Documented Code for datatool v2.32

Nicola L. C. Talbot

http://www.dickimaw-books.com/

2019-09-27

This is the documented code for the datatool bundle. See datatool-user.pdf for the main user manual.

Contents

1 datatool-base.sty

This package provides all the basic commands needed by various packages in the datatool bundle.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-base}[2019/09/27 v2.32 (NLCT)]
```

Required packages:

```
\RequirePackage{etoolbox}
\RequirePackage{amsmath}
\RequirePackage{xkeyval}
\RequirePackage{xfor}
\RequirePackage{ifthen}
```

Version of required that fixes `\su@IfSubStringInString`

```
\RequirePackage{substr}[2009/10/20]
```

1.1 Package Options

verbose Define key for package option verbose. (This also switches the fp messages on/off if datatool-fp used.) This boolean may already have been defined if datatool has been loaded.

```
\ifundef{\ifdtlverbose}
{
  \define@boolkey{datatool-base.sty}[dtl]{verbose}[true]{}
}%
{}
```

math Determine whether to use fp or pgfmath for the arithmetic commands. The default is to use fp.

```
\define@choicekey{datatool-base.sty}{math}[\val\nr]{fp,pgfmath}{%
  \renewcommand*\@dtl@mathprocessor{#1}%
}
```

utf8 Enable UTF-8 support in comparison handlers. This is still a bit experimental, so it needs to be explicitly switched on.

```
\define@boolkey{datatool-base.sty}[@dtl@]{utf8}[true]{}
\ifdef\UTFviii@two@octets
{\booltrue{@dtl@utf8}}%
{\boolfalse{@dtl@utf8}}
```

tlenableUTFviii

```
\newcommand*{\dtlenableUTFviii}{\booltrue{@dtl@utf8}}
```

ldisableUTFviii

```
\newcommand*{\dtldisableUTFviii}{\boolfalse{@dtl@utf8}}
```

l@mathprocessor

```
\providecommand*{\@dtl@mathprocessor}{fp}
```

Process options:

```
\ProcessOptionsX
```

Load package dealing with numerical processes:

```
\RequirePackage{datatool-\@dtl@mathprocessor}
```

1.2 Utilities

\dtl@message

```
\dtl@message{⟨message string⟩}
```

Displays message only if the verbose option is set.

```
\newcommand*{\dtl@message}[1]{%
  \ifdtlverbose\typeout{#1}\fi
}
```

\@dtl@toks

```
\newtoks\@dtl@toks
```

\@dtl@tmpcount Define temporary count register

```
\newcount\@dtl@tmpcount
```

\dtl@tmplength Define temporary length register:

```
\newlength\dtl@tmplength
```

\dtl@ifsingle

```
\dtl@ifsingle{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

If there is only one object in ⟨*arg*⟩ (without expansion) do ⟨*true part*⟩, otherwise do false part.

```
\newcommand{\dtl@ifsingle}[3]{%
  \def\@dtl@arg{#1}%
  \ifdefempty{\@dtl@arg}%
  {%
    #3%
  }%
  {%
    \@dtl@ifsingle#1\@nil{#2}{#3}%
  }%
}
```

\@dtl@ifsingle

```
\def\@dtl@ifsingle#1#2\@nil#3#4{%
  \def\dtl@sg@arg{#2}%
  \ifdefempty{\dtl@sg@arg}%
  {%
   #3%
  }%
  {%
   #4%
  }%
}
```

singleorUTFviii

As above but also checks for UTF8.

```
\newcommand{\dtl@ifsingleorUTFviii}[3]{%
  \ifbool{@dtl@utf8}
  {%
    \def\@dtl@arg{#1}%
    \ifdefempty{\@dtl@arg}%
    {%
      #3%
    }%
    {%
      \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
      {%
        \dtl@getfirst@UTFviii#1\@nil\end@dtl@getfirst@UTFviii
        \ifdefempty\dtl@rest{#2}{#3}%
      }%
      {%
        \@dtl@ifsingle#1\@nil{#2}{#3}%
      }%
    }%
  }%
  {%
    \dtl@ifsingle{#1}{#2}{#3}%
  }%
}%
```

ifintopenbetween

`\dtlifintopenbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}`

If we're dealing with integers it's more efficient to use TEX's `\ifnum`.

```
\newcommand{\dtlifintopenbetween}[5]{%
  \ifnum#1>#2\relax
```

Greater than minimum value. Is it less than the maximum?

```
    \ifnum#1<#3\relax
      #4%
    \else
```

```
      #5%
    \fi
  \else
    #5%
  \fi
}
```

intclosedbetween

```
\dtlifintclosedbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

If we're dealing with integers it's more efficient to use TEX's \ifnum.

```
\newcommand{\dtlifintclosedbetween}[5]{%
  \dtlifintopenbetween{#1}{#2}{#3}{#4}%
  {%
```

Check end points.

```
    \ifnum#1=#2\relax
      #4%
    \else
      \ifnum#1=#3\relax
        #4%
      \else
        #5%
      \fi
    \fi
  }%
}
```

ng@collect@body

Need long versions of 's \collect@body. These macros are adapted from the macros defined by amsmath.

```
\long\def\long@collect@body#1{%
  \@envbody{\@xp#1\@xp{\the\@envbody}}%
  \edef\process@envbody{\the\@envbody\@nx\end{\@currenvir}}%
  \@envbody\@emptytoks \def\begin@stack{b}%
  \begingroup
  \@xp\let\csname\@currenvir\endcsname\long@collect@@body
  \edef\process@envbody{\@xp\@nx\csname\@currenvir\endcsname}%
  \process@envbody
}
```

g@addto@envbody

Adapted from 's \addto@envbody

```
\long\def\long@addto@envbody#1{%
  \toks@{#1}%
  \edef\@dtl@tmp{\the\@envbody\the\toks@}%
  \global\@envbody\@xp{\@dtl@tmp}%
}
```

g@collect@@body Adapted from 's \collect@body

```
\long\def\long@collect@@body#1\end#2{%
  \protected@edef\begin@stack{%
    \long@push@begins#1\begin\end \@xp\@gobble\begin@stack
  }%
  \ifx\@empty\begin@stack
    \endgroup
    \@checkend{#2}%
    \long@addto@envbody{#1}%
  \else
    \long@addto@envbody{#1\end{#2}}%
  \fi
  \process@envbody
}
```

ong@push@begins Adapted from 's \push@begins

```
\long\def\long@push@begins#1\begin#2{%
  \ifx\end#2\else b\@xp\long@push@begins\fi
}
```

1.2.1 General List Utilities

\DTLifinlist

```
\DTLifinlist{⟨element⟩}{⟨list⟩}{⟨true part⟩}{⟨false part⟩}
```

If ⟨*element*⟩ is contained in the comma-separated list given by ⟨*list*⟩, then do ⟨*true part*⟩ otherwise do false part. (Does a one level expansion on ⟨*list*⟩, but no expansion on ⟨*element*⟩.)

```
\newcommand*{\DTLifinlist}[4]{%
  \def\@dtl@doifinlist##1,#1,##2\end@dtl@doifinlist{%
     \def\@before{##1}%
     \def\@after{##2}%
  }%
  \expandafter\@dtl@doifinlist\expandafter,#2,#1,\@nil
    \end@dtl@doifinlist
  \ifx\@after\@nnil
% not found
    #4%
  \else
% found
    #3%
  \fi
}
```

TLlistskipempty If true, skip empty elements in the list-related commands below.

```
\newif\ifDTLlistskipempty
\DTLlistskipemptytrue
```

\DTLlistelement

```
\DTLlistelement{⟨list⟩}{⟨idx⟩}
```

Does the ⟨*idx*⟩th element in the list. The index should start from 1 for the first element.

```
\newrobustcmd{\DTLlistelement}[2]{%
```

The \@for-loop is scoped in case it's nested.

```
  \begingroup
    \@dtl@tmpcount=0\relax
    \@for\@dtl@element:=#1\do{%
     \ifDTLlistskipempty
       \ifdefempty{\@dtl@element}%
       {}%
       {%
         \advance\@dtl@tmpcount by 1\relax%
         \ifnum\@dtl@tmpcount=#2 \@dtl@element\@endfortrue\fi
       }%
     \else
       \advance\@dtl@tmpcount by 1\relax%
       \ifnum\@dtl@tmpcount=#2 \@dtl@element\@endfortrue\fi
     \fi
    }%
    \if@endfor \else\@dtl@listelement@outofrange{#2}\fi
  \endgroup
}
```

fetchlistelement

```
\DTLfetchlistelement{⟨list⟩}{⟨idx⟩}{⟨cs⟩}
```

Fetches the ⟨*idx*⟩th element in the list and stores in ⟨*cs*⟩. The index should start from 1 for the first element.

```
\newrobustcmd{\DTLfetchlistelement}[3]{%
```

The \@for-loop is scoped in case it's nested.

```
 \begingroup
   \@dtl@tmpcount=0\relax
   \@for\@dtl@element:=#1\do{%
    \ifDTLlistskipempty
      \ifdefempty{\@dtl@element}%
      {}%
      {%
        \advance\@dtl@tmpcount by 1\relax%
        \ifnum\@dtl@tmpcount=#2 \@endfortrue\fi
      }%
    \else
      \advance\@dtl@tmpcount by 1\relax%
      \ifnum\@dtl@tmpcount=#2 \@endfortrue\fi
```

```
     \fi
    }%
    \if@endfor \else\def\@dtl@element{\@dtl@listelement@outofrange{#2}}\fi
    \edef\x{%
      \endgroup
      \noexpand\def\noexpand#3{\expandonce\@dtl@element}%
    }\x
}
```

ment@outofrange

```
\newcommand{\@dtl@listelement@outofrange}[1]{%
  \PackageWarning{datatool-base}{List index ‘\number#1’ out of range}%
}
```

TLnumitemsinlist

\DTLnumitemsinlist{⟨*list*⟩}{⟨*cmd*⟩}

Counts number of non-empty elements in list and stores result in control sequence ⟨*cmd*⟩.

```
\newrobustcmd{\DTLnumitemsinlist}[2]{%
  \@dtl@tmpcount=0\relax
  \@for\@dtl@element:=#1\do{%
     \ifDTLlistskipempty
       \ifdefempty{\@dtl@element}%
       {}%
       {\advance\@dtl@tmpcount by 1\relax}%
     \else
       \advance\@dtl@tmpcount by 1\relax
     \fi
  }%
  \edef#2{\number\@dtl@tmpcount}%
}
```

\dtl@choplast

\dtl@choplast{⟨*list*⟩}{⟨*rest*⟩}{⟨*last*⟩}

Chops the last element off a comma separated list, putting the last element in the control sequence ⟨*last*⟩ and putting the rest in the control sequence ⟨*rest*⟩. The control sequence ⟨*list*⟩ is unchanged. If the list is empty, both ⟨*last*⟩ and ⟨*rest*⟩ will be empty.

```
\newcommand*{\dtl@choplast}[3]{%
```

Set ⟨*rest*⟩ to empty:

```
  \let#2\@empty
```

Set ⟨*last*⟩ to empty:

```
  \let#3\@empty
```

Iterate through ⟨*list*⟩:

```
  \@for\@dtl@element:=#1\do{%
    \ifdefempty{#3}%
    {%
```

First iteration, don't set ⟨*rest*⟩.

```
    }%
    {%
      \ifdefempty{#2}%
      {%
```

Second iteration, set ⟨*rest*⟩ to ⟨*last*⟩ (which is currently set to the previous value:

```
        \expandafter\toks@\expandafter{#3}%
        \edef#2{{\the\toks@}}%
      }%
      {%
```

Subsequent iterations, set ⟨*rest*⟩ to ⟨*rest*⟩,⟨*last*⟩ (⟨*last*⟩ is currently set to the previous value):

```
        \expandafter\toks@\expandafter{#3}%
        \expandafter\@dtl@toks\expandafter{#2}%
        \edef#2{\the\@dtl@toks,{\the\toks@}}%
      }%
    }%
```

Now set ⟨*last*⟩ to current element.

```
    \let#3=\@dtl@element%
  }%
}
```

\dtl@chopfirst

```
\dtl@chopfirst{⟨list⟩}{⟨first⟩}{⟨rest⟩}
```

Chops first element off ⟨*list*⟩ and store in ⟨*first*⟩. The remainder of the list is stored in ⟨*rest*⟩. (⟨*list*⟩ remains unchanged.)

```
\newcommand*{\dtl@chopfirst}[3]{%
  \let#2=\@empty
  \let#3=\@empty
  \@for\@dtl@element:=#1\do{%
    \let#2=\@dtl@element
    \@endfortrue
  }%
  \if@endfor
   \let#3=\@forremainder
  \fi
  \@endforfalse
}
```

I made a bit of a blunder here. `\dtl@sortlist` was supposed to work with commands like `\dtlcompare`, but those commands require three arguments, the first being the register in which to store the result. This contradicts the requirements of `\dtl@sortlist`. The "bug fix" in v2.26 fixed it to work with commands like `\dtlcompare`, but that broke the documented design (which breaks the glossaries package). The other problem is that `\dtl@insertinto` actually sorts in the reverse order. So v2.27 undoes the change from v2.26 to ensure backward compatibility and provides an alternative user-level command `\dltsortlist`, that's designed to work with the three-argument handler commands like `\dtlcompare`.

`\dtl@sortlist` `\dtl@sortlist{⟨list⟩}{⟨criteria cmd⟩}`

Performs an insertion sort on ⟨*list*⟩, where ⟨*criteria cmd*⟩ is a macro which takes two arguments ⟨*a*⟩ and ⟨*b*⟩. ⟨*criteria cmd*⟩ must set the count register `\dtl@sortresult` to either −1 (⟨*b*⟩ less than ⟨*a*⟩), 0 (⟨*a*⟩ is equal to ⟨*b*⟩) or 1 (⟨*b*⟩ is greater than ⟨*a*⟩.)

```
\newcommand{\dtl@sortlist}[2]{%
\def\@dtl@sortedlist{}%
\@for\@dtl@currentrow:=#1\do{%
\expandafter\dtl@insertinto\expandafter
  {\@dtl@currentrow}{\@dtl@sortedlist}{#2}%
\@endforfalse}%
\let#1=\@dtl@sortedlist
}
```

`\dtl@insertinto` `\dtl@insertinto{⟨element⟩}{⟨sorted-list⟩}{⟨criteria cmd⟩}`

Inserts ⟨*element*⟩ into the sorted list ⟨*sorted-list*⟩ according to the criteria given by ⟨*criteria cmd*⟩ (see above.)

```
\newcommand{\dtl@insertinto}[3]{%
  \def\@dtl@newsortedlist{}%
  \@dtl@insertdonefalse
  \@for\dtl@srtelement:=#2\do{%
    \if@dtl@insertdone
      \expandafter\toks@\expandafter{\dtl@srtelement}%
      \edef\@dtl@newstuff{{\the\toks@}}%
    \else

      \expandafter#3\expandafter{\dtl@srtelement}{#1}%

      \ifnum\dtl@sortresult<0\relax
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \@dtl@toks{#1}%
        \edef\@dtl@newstuff{{\the\@dtl@toks},{\the\toks@}}%
        \@dtl@insertdonetrue
      \else
        \expandafter\toks@\expandafter{\dtl@srtelement}%
    \edef\@dtl@newstuff{{\the\toks@}}%
   \fi
  \fi
  \ifdefempty{\@dtl@newsortedlist}%
  {%
   \expandafter\toks@\expandafter{\@dtl@newstuff}%
   \edef\@dtl@newsortedlist{\the\toks@}%
  }%
  {%
   \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
   \expandafter\@dtl@toks\expandafter{\@dtl@newstuff}%
   \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
  }%
  \@endforfalse
 }%
 \ifdefempty{\@dtl@newsortedlist}%
 {%
  \@dtl@toks{#1}%
  \edef\@dtl@newsortedlist{{\the\@dtl@toks}}%
 }%
 {%
  \if@dtl@insertdone
  \else
   \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
   \@dtl@toks{#1}%
   \edef\@dtl@newsortedlist{\the\toks@,{\the\@dtl@toks}}%
  \fi
 }%
 \global\let#2=\@dtl@newsortedlist
}
```

\dtlsortlist

\dtlsortlist{⟨*list*⟩}{⟨*criteria cmd*⟩}

As \dtl@sortlist but the ⟨*criteria*⟩ command must take three arguments.

```
\newcommand{\dtlsortlist}[2]{%
\def\@dtl@sortedlist{}%
\@for\@dtl@currentrow:=#1\do{%
\expandafter\dtlinsertinto\expandafter
 {\@dtl@currentrow}{\@dtl@sortedlist}{#2}%
\@endforfalse}%
\let#1=\@dtl@sortedlist
}
```

\dtlinsertinto

\dtlinsertinto{⟨*element*⟩}{⟨*sorted-list*⟩}{⟨*criteria cmd*⟩}

Inserts ⟨*element*⟩ into the sorted list ⟨*sorted-list*⟩ according to the criteria given by ⟨*criteria cmd*⟩, which should be a command that takes three arguments {⟨*reg*⟩}{⟨*A*⟩}{⟨*B*⟩}, where ⟨*reg*⟩ is a count register in which to store the result, ⟨*A*⟩ is the first element and ⟨*B*⟩ is the second element to compare.

```
\newcommand{\dtlinsertinto}[3]{%
  \def\@dtl@newsortedlist{}%
  \@dtl@insertdonefalse
  \@for\dtl@srtelement:=#2\do{%
    \expandafter\DTLifSubString\expandafter{\dtl@srtelement}{,}
    {%
      \expandafter\toks@\expandafter{\dtl@srtelement}%
      \edef\dtl@srtelement{{\the\toks@}}%
    }%
    {%
    }
    \if@dtl@insertdone
      \let\@dtl@newstuff\dtl@srtelement
    \else
      \expandafter#3\expandafter\dtl@sortresult
        \expandafter{\dtl@srtelement}{#1}%
      \ifnum\dtl@sortresult>0\relax
        \DTLifSubString{#1}{,}%
        {%
          \@dtl@toks{{#1}}%
        }%
        {%
          \@dtl@toks{#1}%
        }%
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{\the\@dtl@toks,\the\toks@}%
        \@dtl@insertdonetrue
      \else
        \expandafter\toks@\expandafter{\dtl@srtelement}%
        \edef\@dtl@newstuff{{\the\toks@}}%
        \let\@dtl@newstuff\dtl@srtelement
      \fi
    \fi
    \ifdefempty{\@dtl@newsortedlist}%
    {%
      \expandafter\toks@\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@}%
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
      \expandafter\@dtl@toks\expandafter{\@dtl@newstuff}%
      \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
    }%
    \@endforfalse
  }%
```

```
    \ifdefempty{\@dtl@newsortedlist}%
    {%
      \DTLifSubString{#1}{,}%
      {%
        \@dtl@toks{{#1}}%
      }%
      {%
        \@dtl@toks{#1}%
      }%
      \edef\@dtl@newsortedlist{\the\@dtl@toks}%
    }%
    {%
      \if@dtl@insertdone
      \else
        \DTLifSubString{#1}{,}%
        {%
          \@dtl@toks{{#1}}%
        }%
        {%
          \@dtl@toks{#1}%
        }%
        \expandafter\toks@\expandafter{\@dtl@newsortedlist}%
        \edef\@dtl@newsortedlist{\the\toks@,\the\@dtl@toks}%
      \fi
    }%
    \global\let#2=\@dtl@newsortedlist
  }
```

\edtlinsertinto

\edtlinsertinto{⟨*element*⟩}{⟨*sorted-list*⟩}{⟨*criteria cmd*⟩}

First expands ⟨*element*⟩ before inserting into the list.

```
  \newcommand*{\edtlinsertinto}[3]{%
    \protected@edef\dtl@srtelement{#1}%
    \expandafter\dtlinsertinto\expandafter{\dtl@srtelement}{#2}{#3}%
  }
```

@dtl@insertdone Define conditional to indicate whether the new entry has been inserted into the sorted list.

```
  \newif\if@dtl@insertdone
```

\dtl@sortresult Define `\dtl@sortresult` to be set by comparison macro.

```
  \newcount\dtl@sortresult
```

This next command is based on the list iteration exercise given at http://www.dickimaw-books.com/latex/admin/html/foreachtips.shtml#oxfordcomma It's designed to format a comma-separated list using `\DTLlistformatsep` between each item except for the last. The separator for the last pair is `\DTLlistformatlastsep` if the list only contains two items or

\DTLlistformatoxford\DTLlistformatlastsep if the list contains three or more items. Each item is formatted according to \DTLlistformatitem.

TLlistformatsep

```
\newcommand*{\DTLlistformatsep}{, }
```

istformatoxford

```
\newcommand*{\DTLlistformatoxford}{}
```

\DTLandname

```
\ifdef\andname
{\newcommand*{\DTLandname}{\andname}}
{\newcommand*{\DTLandname}{\&}}
```

stformatlastsep

```
\newcommand*{\DTLlistformatlastsep}{ \DTLandname\space}
```

Llistformatitem

```
\newcommand*{\DTLlistformatitem}[1]{#1}
```

matlist@handler

```
\newcommand*{\@dtl@formatlist@handler}[1]{%
  \@dtl@formatlist@itemsep
  \@dtl@formatlist@lastitem
  \renewcommand{\@dtl@formatlist@lastitem}{%
    \renewcommand{\@dtl@formatlist@itemsep}{%
      \DTLlistformatsep
      \renewcommand*{\@dtl@formatlist@prelastitemsep}{%
        \DTLlistformatoxford}}%
    \renewcommand{\@dtl@formatlist@prelastitem}{%
      \@dtl@formatlist@prelastitemsep
       \DTLlistformatlastsep}%
    \DTLlistformatitem{#1}%
  }%
}%
```

\DTLformatlist Formats the comma-separated list supplied in its argument. The unstarred version adds grouping.

```
\newrobustcmd*{\DTLformatlist}{%
 \@ifstar{\s@dtlformatlist}{\@dtlformatlist}%
}
```

s@dtlformatlist Starred version of \DTLformatlist doesn't add grouping.

```
\newcommand*{\s@dtlformatlist}[1]{%
  \def\@dtl@formatlist@itemsep{}%
  \def\@dtl@formatlist@lastitem{}%
  \def\@dtl@formatlist@prelastitem{}%
  \def\@dtl@formatlist@prelastitemsep{}%
  \@for\@dtl@formatlist@item:=#1\do{%
    \ifDTLlistskipempty
      \ifdefempty{\@dtl@formatlist@item}%
      {}%
      {\expandafter\@dtl@formatlist@handler\expandafter{\@dtl@formatlist@item}}%
    \else
      \expandafter\@dtl@formatlist@handler\expandafter{\@dtl@formatlist@item}%
    \fi
  }%
  \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
}
```

\@dtlformatlist Unstarred version of \DTLformatlist adds grouping.

```
\newcommand*{\@dtlformatlist}[1]{{\s@dtlformatlist{#1}}}
```

1.2.2 General Token Utilities

ks@gput@right@cx

`\dtl@toks@gput@right@cx{⟨toks name⟩}{⟨stuff⟩}`

Globally appends stuff to token register \⟨*toks name*⟩

```
\newcommand{\@dtl@toks@gput@right@cx}[2]{%
  \def\@dtl@toks@name{#1}%
  \edef\@dtl@stuff{#2}%
  \global\csname\@dtl@toks@name\endcsname\expandafter
    \expandafter\expandafter{\expandafter\the
    \csname\expandafter\@dtl@toks@name\expandafter\endcsname\@dtl@stuff}%
}
```

concat@middle@cx

`\dtl@toks@gconcat@middle@cx{⟨toks name⟩}{⟨before toks⟩}{⟨stuff⟩}{⟨after toks⟩}`

Globally sets token register \⟨*toks name*⟩ to the contents of ⟨*before toks*⟩ concatenated with ⟨*stuff*⟩ (expanded) and the contents of ⟨*after toks*⟩

```
\newcommand{\@dtl@toks@gconcat@middle@cx}[4]{%
  \def\@dtl@toks@name{#1}%
  \edef\@dtl@stuff{#3}%
  \global\csname\@dtl@toks@name\endcsname\expandafter\expandafter
    \expandafter\expandafter\expandafter
    \expandafter\expandafter{\expandafter\expandafter\expandafter
    \the\expandafter\expandafter\expandafter#2%
    \expandafter\@dtl@stuff\the#4}%
}
```

1.3 Locale Dependent Information

@numgrpsepcount

Define count register to count the digits between the number group separators.

```
\newcount\@dtl@numgrpsepcount
```

\@dtl@decimal

The current decimal character is stored in `\@dtl@decimal`.

```
\newcommand*{\@dtl@decimal}{.}
```

numbergroupchar

The current number group character is stored in `\@dtl@numbergroupchar`.

```
\newcommand*{\@dtl@numbergroupchar}{,}
```

TLsetnumberchars

`\DTLsetnumberchars{⟨number group char⟩}{⟨decimal char⟩}`

This sets the decimal character and number group characters.

```
\newcommand*{\DTLsetnumberchars}[2]{%
  \renewcommand*{\@dtl@numbergroupchar}{#1}%
  \renewcommand*{\@dtl@decimal}{#2}%
  \@dtl@construct@getnums
  \@dtl@construct@stripnumgrpchar{#1}%
}
```

truct@getintfrac

`\@dtl@construct@getintfrac{⟨char⟩}`

This constructs the macros for extracting integer and fractional parts from a real number using the decimal character ⟨*char*⟩.

converttodecimal

`\DTLconverttodecimal{⟨num⟩}{⟨cmd⟩}`

`\DTLconverttodecimal` will convert locale dependent ⟨*num*⟩ a decimal number in a form that can be used in the macros defined in the fp package. The resulting number is stored in ⟨*cmd*⟩. This command has to be redefined whenever the decimal and number group characters are changed as they form part of the command definitions.

```
\edef\@dtl@construct@getintfrac#1{%
  \noexpand\def\noexpand\@dtl@getintfrac##1#1##2\noexpand\relax{%
    \noexpand\@dtl@get@intpart{##1}%
    \noexpand\def\noexpand\@dtl@fracpart{##2}%
    \noexpand\ifdefempty{\noexpand\@dtl@fracpart}
    {%
      \noexpand\def\noexpand\@dtl@fracpart{0}%
    }%
    {%
```

```
      \noexpand\@dtl@getfracpart##2\noexpand\relax
      \noexpand\@dtl@choptrailingzeroes{\noexpand\@dtl@fracpart}%
    }%
  }%
  \noexpand\def\noexpand\@dtl@getfracpart##1#1\noexpand\relax{%
    \noexpand\def\noexpand\@dtl@fracpart{##1}%
  }%
  \noexpand\def\noexpand\DTLconverttodecimal##1##2{%
    \noexpand\dtl@ifsingle{##1}%
    {%
      \noexpand\expandafter\noexpand\toks@\noexpand\expandafter{##1}%
      \noexpand\edef\noexpand\@dtl@tmp{\noexpand\the\noexpand\toks@}%
    }%
    {%
      \noexpand\def\noexpand\@dtl@tmp{##1}%
    }%
    \noexpand\@dtl@standardize@currency\noexpand\@dtl@tmp
    \noexpand\ifdefempty{\noexpand\@dtl@org@currency}%
    {%
    }%
    {%
      \noexpand\let\noexpand\@dtl@currency\noexpand\@dtl@org@currency
    }%
    \noexpand\expandafter
      \noexpand\@dtl@getintfrac\noexpand\@dtl@tmp#1\noexpand\relax
    \noexpand\edef##2{\noexpand\@dtl@intpart.\noexpand\@dtl@fracpart}%
  }%
}
```

nstruct@getnums The following calls the above with the relevant decimal character:

```
\newcommand*{\@dtl@construct@getnums}{%
  \expandafter\@dtl@construct@getintfrac\expandafter{\@dtl@decimal}%
}
```

dtl@get@intpart The following gets the integer part (adjusting for repeating +/- signs if necessary.) Sets \@dtl@intpart.

```
\newcommand*{\@dtl@get@intpart}[1]{%
  \@dtl@tmpcount=1\relax
  \def\@dtl@intpart{#1}%
  \ifx\@dtl@intpart\@empty
    \def\@dtl@intpart{0}%
  \else
    \def\@dtl@intpart{}%
    \@dtl@get@int@part#1.\relax%
  \fi
  \ifnum\@dtl@tmpcount<0\relax
    \edef\@dtl@intpart{-\@dtl@intpart}%
  \fi
  \@dtl@strip@numgrpchar{\@dtl@intpart}%
}
```

tl@get@int@part

```
\def\@dtl@get@int@part#1#2\relax{%
  \def\@dtl@argi{#1}%
  \def\@dtl@argii{#2}%
  \ifx\protect#1\relax%
    \let\@dtl@get@nextintpart=\@dtl@get@int@part
  \else
    \expandafter\ifx\@dtl@argi\$%
      \let\@dtl@get@nextintpart=\@dtl@get@int@part
    \else
      \ifx-#1%
        \multiply\@dtl@tmpcount by -1\relax
        \let\@dtl@get@nextintpart=\@dtl@get@int@part
      \else
        \if\@dtl@argi+%
          \let\@dtl@get@nextintpart=\@dtl@get@int@part
        \else
          \def\@dtl@intpart{#1}%
          \ifx.\@dtl@argii
            \let\@dtl@get@nextintpart=\@gobble
          \else
            \let\@dtl@get@nextintpart=\@dtl@get@next@intpart
          \fi
        \fi
      \fi
    \fi
  \fi
  \@dtl@get@nextintpart#2\relax
}
```

et@next@intpart

```
\def\@dtl@get@next@intpart#1.\relax{%
  \edef\@dtl@intpart{\@dtl@intpart#1}%
}
```

optrailingzeroes

\@dtl@choptrailingzeroes{⟨*cmd*⟩}

Chops trailing zeroes from number given by ⟨*cmd*⟩.

```
\newcommand*{\@dtl@choptrailingzeroes}[1]{%
  \def\@dtl@tmpcpz{}%
  \expandafter\@dtl@chop@trailingzeroes#1\@nil%
  \let#1=\@dtl@tmpcpz
}
```

@trailingzeroes

Trailing zeroes are chopped using a recursive algorithm. \@dtl@tmpcpz needs to be set before using this. (The chopped number is put in this control sequence.)

```
\def\@dtl@chop@trailingzeroes#1#2\@nil{%
  \dtlifnumeq{#2}{0}%
  {%
    \edef\@dtl@tmpcpz{\@dtl@tmpcpz#1}%
    \let\@dtl@chopzeroesnext=\@dtl@gobbletonil
  }%
  {%
    \edef\@dtl@tmpcpz{\@dtl@tmpcpz#1}%
    \let\@dtl@chopzeroesnext=\@dtl@chop@trailingzeroes
  }%
  \@dtl@chopzeroesnext#2\@nil
}
```

No-op macro to end recursion:

dtl@gobbletonil

```
\def\@dtl@gobbletonil#1\@nil{}
```

@truncatedecimal

\dtl@truncatedecimal⟨*cmd*⟩

Truncates decimal given by ⟨*cmd*⟩ to an integer (assumes the number is in decimal format with full stop as decimal point.)

```
\newcommand*{\dtl@truncatedecimal}[1]{%
  \expandafter\@dtl@truncatedecimal#1.\@nil#1%
}
```

truncatedecimal

```
\def\@dtl@truncatedecimal#1.#2\@nil#3{%
  \def#3{#1}%
}
```

strip@numgrpchar

\@dtl@strip@numgrpchar{⟨*cmd*⟩}

Strip the number group character from the number given by ⟨*cmd*⟩.

```
\newcommand*{\@dtl@strip@numgrpchar}[1]{%
  \def\@dtl@stripped{}%
  \edef\@dtl@do@stripnumgrpchar{%
    \noexpand\@@dtl@strip@numgrpchar#1\@dtl@numbergroupchar
    \noexpand\relax
  }%
  \@dtl@do@stripnumgrpchar
  \let#1=\@dtl@stripped
}
```

stripnumgrpchar

The following macro constructs \@@dtl@strip@numgrpchar.

```
\edef\@dtl@construct@stripnumgrpchar#1{%
  \noexpand\def\noexpand\@@dtl@strip@numgrpchar##1#1##2\noexpand\relax{%
    \noexpand\expandafter\noexpand\toks@\noexpand\expandafter
      {\noexpand\@dtl@stripped}%
    \noexpand\edef\noexpand\@dtl@stripped{%
      \noexpand\the\noexpand\toks@
      ##1%
    }%
    \noexpand\def\noexpand\@dtl@tmp{##2}%
    \noexpand\ifx\noexpand\@dtl@tmp\noexpand\@empty
      \noexpand\let\noexpand\@dtl@next=\noexpand\relax
    \noexpand\else
      \noexpand\let\noexpand\@dtl@next=\noexpand\@@dtl@strip@numgrpchar
    \noexpand\fi
    \noexpand\@dtl@next##2\noexpand\relax
  }%
}
```

Ldecimaltolocale

\DTLdecimaltolocale{⟨*number*⟩}{⟨*cmd*⟩}

Define command to convert a decimal number into the locale dependent format. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLdecimaltolocale}[2]{%
  \edef\@dtl@tmpdtl{#1}%
  \expandafter\@dtl@decimaltolocale\@dtl@tmpdtl.\relax
  \dtlifnumeq{\@dtl@fracpart}{0}%
  {%
    \edef#2{\@dtl@intpart}%
  }%
  {%
    \edef#2{\@dtl@intpart\@dtl@decimal\@dtl@fracpart}%
  }%
}
```

decimaltolocale

Convert the integer part (store in \@dtl@intpart)

```
\def\@dtl@decimaltolocale#1.#2\relax{%
  \@dtl@decimaltolocaleint{#1}%
  \def\@dtl@fracpart{#2}%
  \ifdefempty\@dtl@fracpart
  {%
    \def\@dtl@fracpart{0}%
  }%
  {%
    \@dtl@decimaltolocalefrac#2\relax
  }%
}
```

imaltolocaleint

```
\def\@dtl@decimaltolocaleint#1{%
  \@dtl@tmpcount=0\relax
  \@dtl@countdigits#1.\relax
  \@dtl@numgrpsepcount=\@dtl@tmpcount\relax
  \divide\@dtl@numgrpsepcount by 3\relax
  \multiply\@dtl@numgrpsepcount by 3\relax
  \advance\@dtl@numgrpsepcount by -\@dtl@tmpcount\relax
  \ifnum\@dtl@numgrpsepcount<0\relax
    \advance\@dtl@numgrpsepcount by 3\relax
  \fi
  \def\@dtl@intpart{}%
  \@dtl@decimal@to@localeint#1.\relax
}
```

dtl@countdigits Counts the number of digits until #2 is a full stop. (increments \@dtl@tmpcount.)

```
\def\@dtl@countdigits#1#2\relax{%
  \advance\@dtl@tmpcount by 1\relax
  \ifx.#2\relax
    \let\@dtl@countnext=\@gobble
  \else
    \let\@dtl@countnext=\@dtl@countdigits
  \fi
  \@dtl@countnext#2\relax
}
```

al@to@localeint

```
\def\@dtl@decimal@to@localeint#1#2\relax{%
  \advance\@dtl@numgrpsepcount by 1\relax
  \ifx.#2\relax
    \edef\@dtl@intpart{\@dtl@intpart#1}%
    \let\@dtl@localeintnext=\@gobble
  \else
    \ifnum\@dtl@numgrpsepcount=3\relax
      \edef\@dtl@intpart{\@dtl@intpart#1\@dtl@numbergroupchar}%
      \@dtl@numgrpsepcount=0\relax
    \else
      \ifnum\@dtl@numgrpsepcount>3\relax
        \@dtl@numgrpsepcount=0\relax
      \fi
      \edef\@dtl@intpart{\@dtl@intpart#1}%
    \fi
    \let\@dtl@localeintnext=\@dtl@decimal@to@localeint
  \fi
```

```
    \@dtl@localeintnext#2\relax
  }
```

maltolocalefrac

Convert the fractional part (store in `\@dtl@fracpart`). `\@dtl@choptrailingzeroes` was originally used, but this interfered with `\dtlround`. Removing `\@dtl@choptrailingzeroes` caused a ‘Number too big’ error, so any fractional part over 2147483647 needs to be trimmed. Unfortunately `\ifnum#1>2147483647` also causes the ‘Number too big’ error, so count the digits, and if the digit count exceeds 9, trim the excess.

```
  \def\@dtl@decimaltolocalefrac#1.\relax{%
    \count@=0\relax
    \@dtl@digitcount#1\relax
    \ifnum\count@>9\relax
      \@dtl@chopexcessfrac#1000000000\@nil
    \else
      \def\@dtl@fracpart{#1}%
    \fi
  }
```

@chopexcessfrac

Chop fractional part to just 9 digits.

```
  \newcommand*{\@dtl@chopexcessfrac}[9]{%
    \def\@dtl@fracpart{#1#2#3#4#5#6#7#8#9}%
    \@dtl@gobbletonil
  }
```

@dtl@digitcount

Counts the number of digits in #1.

```
  \newcommand*{\@dtl@digitcount}[1]{%
    \ifx\relax#1\relax
      \let\@dtl@digitcountnext\relax
    \else
      \advance\count@ by \@ne
      \let\@dtl@digitcountnext\@dtl@digitcount
    \fi
    \@dtl@digitcountnext
  }
```

ecimaltocurrency

`\DTLdecimaltocurrency{`⟨*number*⟩`}{`⟨*cmd*⟩`}`

This converts a decimal number into the locale dependent currency format. Stores result in ⟨*cmd*⟩ which must be a control sequence.

```
  \newcommand*{\DTLdecimaltocurrency}[2]{%
    \edef\@dtl@tmpdtl{#1}%
    \expandafter\@dtl@decimaltolocale\@dtl@tmpdtl.\relax
    \dtl@truncatedecimal\@dtl@tmpdtl
    \@dtl@tmpcount=\@dtl@tmpdtl\relax
    \expandafter\@dtl@toks\expandafter{\@dtl@currency}%
    \dtlifnumeq{\@dtl@fracpart}{0}%
    {%
      \ifnum\@dtl@tmpcount<0\relax
        \@dtl@tmpcount = -\@dtl@tmpcount\relax
        \edef#2{-\the\@dtl@toks\the\@dtl@tmpcount\@dtl@decimal00}%
      \else
        \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal00}%
      \fi
    }%
    {%
      \ifnum\@dtl@tmpcount<0\relax
        \@dtl@tmpcount = -\@dtl@tmpcount\relax
        \ifnum\@dtl@fracpart<10\relax
          \edef#2{%
            -\the\@dtl@toks\number\@dtl@tmpcount
            \@dtl@decimal\@dtl@fracpart0%
          }%
        \else
          \edef#2{%
            -\the\@dtl@toks\number\@dtl@tmpcount
            \@dtl@decimal\@dtl@fracpart
          }%
        \fi
      \else
        \ifnum\@dtl@fracpart<10\relax
          \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal\@dtl@fracpart0}%
        \else
          \edef#2{\the\@dtl@toks\@dtl@intpart\@dtl@decimal\@dtl@fracpart}%
        \fi
      \fi
    }%
  }
```

Set the defaults:

```
\@dtl@construct@getnums
\expandafter\@dtl@construct@stripnumgrpchar\expandafter
  {\@dtl@numbergroupchar}
```

1.3.1 Currencies

@dtl@currencies \@dtl@currencies stores all known currencies.

```
\newcommand*{\@dtl@currencies}{\$,\pounds}
```

ewcurrencysymbol

\DTLaddcurrency{⟨*symbol*⟩}

Adds ⟨*symbol*⟩ to the list of known currencies

```
\newcommand*{\DTLnewcurrencysymbol}[1]{%
  \expandafter\toks@\expandafter{\@dtl@currencies}%
  \@dtl@toks{#1}%
  \edef\@dtl@currencies{\the\@dtl@toks,\the\toks@}%
}
```

If any of the following currency commands have been defined, add them to the list:

```
\AtBeginDocument{%
  \@ifundefined{texteuro}{}{\DTLnewcurrencysymbol{\texteuro}}%
  \@ifundefined{textdollar}{}{\DTLnewcurrencysymbol{\textdollar}}%
  \@ifundefined{textstirling}{}{\DTLnewcurrencysymbol{\textstirling}}%
  \@ifundefined{textyen}{}{\DTLnewcurrencysymbol{\textyen}}%
  \@ifundefined{textwon}{}{\DTLnewcurrencysymbol{\textwon}}%
  \@ifundefined{textcurrency}{}{\DTLnewcurrencysymbol{\textcurrency}}%
  \@ifundefined{euro}{}{\DTLnewcurrencysymbol{\euro}}%
  \@ifundefined{yen}{}{\DTLnewcurrencysymbol{\yen}}%
}
```

dardize@currency `\@dtl@standardize@currency{⟨cmd⟩}`

Substitutes the first currency symbol found in ⟨*cmd*⟩ with `\$`. This is used when testing text to determine if it is currency. The original currency symbol is stored in `\@dtl@org@currency`, so that it can be replaced later. If no currency symbol is found, `\@dtl@org@currency` will be empty.

```
\newcommand{\@dtl@standardize@currency}[1]{%
  \def\@dtl@org@currency{}%
  \@for\@dtl@thiscurrency:=\@dtl@currencies\do{%
    \expandafter\toks@\expandafter{\@dtl@thiscurrency}%
    \edef\@dtl@dosubs{\noexpand\DTLsubstitute{\noexpand#1}%
     {\the\toks@}{\noexpand\$}}%
    \@dtl@dosubs
    \ifdefempty{\@dtl@replaced}%
    {%
    }%
    {%
      \let\@dtl@org@currency=\@dtl@replaced
      \@endfortrue
    }%
  }%
  \@endforfalse
}
```

\@dtl@currency `\@dtl@currency` is set by `\DTLlocaltodecimal` and `\@dtl@checknumerical`. It is used by `\DTLdecimaltocurrency`. Set to `\$` by default.

```
\newcommand*{\@dtl@currency}{\$}
```

defaultcurrency \DTLsetdefaultcurrency{⟨*symbol*⟩} sets the default currency.

```
\newcommand*{\DTLsetdefaultcurrency}[1]{%
  \renewcommand*{\@dtl@currency}{#1}%
}
```

1.4 Floating Point Arithmetic

The commands defined in this section all use the equivalent commands provided by the fp or pgfmath packages, but first convert the decimal number into the required format.

\DTLadd

```
\DTLadd{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}
```

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ + ⟨*num2*⟩

```
\newcommand*{\DTLadd}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtladd{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
```

\DTLgadd Global version

```
\newcommand*{\DTLgadd}[3]{%
  \DTLadd{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
```

\DTLaddall

```
\DTLaddall{⟨cmd⟩}{⟨num list⟩}
```

Sums all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLaddall}[2]{%
  \def\@dtl@sum{0}%
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \dtladd{\@dtl@sum}{\@dtl@sum}{\@dtl@num}%
  }%
  \ifdefempty{\@dtl@replaced}%
```

```
    {%
      \DTLdecimaltolocale{\@dtl@sum}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@sum}{#1}%
    }%
  }
```

\DTLgaddall

`\DTLgaddall{⟨cmd⟩}{⟨num list⟩}`

Global version

```
  \newcommand*{\DTLgaddall}[2]{%
    \DTLaddall{\@dtl@tmpi}{#2}%
    \global\let#1=\@dtl@tmpi
  }
```

\DTLsub

`\DTLsub{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}`

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ - ⟨*num2*⟩

```
  \newcommand*{\DTLsub}[3]{%
    \DTLconverttodecimal{#2}{\@dtl@numi}%
    \DTLconverttodecimal{#3}{\@dtl@numii}%
    \dtlsub{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \DTLdecimaltolocale{\@dtl@tmp}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
    }%
  }
```

\DTLgsub Global version

```
  \newcommand*{\DTLgsub}[3]{%
    \DTLsub{\@dtl@tmpii}{#2}{#3}%
    \global\let#1=\@dtl@tmpii
  }
```

\DTLmul

`\DTLmul{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}`

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ × ⟨*num2*⟩

```
\newcommand*{\DTLmul}[3]{%
  \let\@dtl@thisreplaced=\@empty
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \dtlmul{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \ifdefempty{\@dtl@thisreplaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
```

\DTLgmul Global version

```
\newcommand*{\DTLgmul}[3]{%
 \DTLmul{\@dtl@tmpii}{#2}{#3}%
 \global\let#1=\@dtl@tmpii
}
```

\DTLdiv `\DTLdiv{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}`

Sets ⟨*cmd*⟩ = ⟨*num1*⟩ / ⟨*num2*⟩

```
\newcommand*{\DTLdiv}[3]{%
  \let\@dtl@thisreplaced=\@empty
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \let\@dtl@thisreplaced=\@dtl@replaced
  }%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtldiv{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
```

```
    \ifdefempty{\@dtl@thisreplaced}%
    {%
      \DTLdecimaltolocale{\@dtl@tmp}{#1}%
    }%
    {%
      \ifdefequal{\@dtl@thisreplaced}{\@dtl@replaced}%
      {%
        \DTLdecimaltolocale{\@dtl@tmp}{#1}%
      }%
      {%
        \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
      }%
    }%
  }
```

\DTLgdiv Global version

```
  \newcommand*{\DTLgdiv}[3]{%
    \DTLdiv{\@dtl@tmpii}{#2}{#3}%
    \global\let#1=\@dtl@tmpii
  }
```

\DTLabs

\DTLabs{⟨*cmd*⟩}{⟨*num*⟩}

Sets ⟨*cmd*⟩ = abs(⟨*num*⟩)

```
  \newcommand*{\DTLabs}[2]{%
    \DTLconverttodecimal{#2}{\@dtl@numi}%
    \dtlabs{\@dtl@tmp}{\@dtl@numi}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \DTLdecimaltolocale{\@dtl@tmp}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
    }%
  }
```

\DTLgabs Global version

```
  \newcommand*{\DTLgabs}[2]{%
    \DTLabs{\@dtl@tmpii}{#2}%
    \global\let#1=\@dtl@tmpii
  }
```

\DTLneg

\DTLneg{⟨*cmd*⟩}{⟨*num*⟩}

Sets ⟨*cmd*⟩ = -⟨*num*⟩

```
\newcommand*{\DTLneg}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlneg{\@dtl@tmp}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
```

\DTLgneg Global version

```
\newcommand*{\DTLgneg}[2]{%
  \DTLneg{\@dtl@tmpii}{#2}%
  \global\let#1=\@dtl@tmpii
}
```

\DTLsqrt

```
\DTLsqrt{⟨cmd⟩}{⟨num⟩}
```

Sets ⟨*cmd*⟩ = sqrt(⟨*num*⟩)

```
\newcommand*{\DTLsqrt}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlroot{\@dtl@tmpi}{\@dtl@numi}{2}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmpi}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmpi}{#1}%
  }%
}
```

\DTLgsqrt Global version

```
\newcommand*{\DTLgsqrt}[2]{%
\DTLsqrt{\@dtl@tmpii}{#2}%
\global\let#1=\@dtl@tmpii
}
```

\DTLmin

```
\DTLmin{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}
```

Sets ⟨*cmd*⟩ = min(⟨*num1*⟩, ⟨*num2*⟩)

```
\newcommand*{\DTLmin}[3]{%
 \DTLconverttodecimal{#2}{\@dtl@numi}%
 \DTLconverttodecimal{#3}{\@dtl@numii}%
 \dtlifnumlt{\@dtl@numi}{\@dtl@numii}%
 {%
   \dtl@ifsingle{#2}%
   {\let#1=#2}%
   {\def#1{#2}}%
 }%
 {%
  \dtl@ifsingle{#3}%
  {\let#1=#3}%
  {\def#1{#3}}%
 }%
}
```

\DTLgmin Global version

```
\newcommand*{\DTLgmin}[3]{%
  \DTLmin{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
```

\DTLminall

\DTLminall{⟨*cmd*⟩}{⟨*num list*⟩}

Finds the minimum value in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLminall}[2]{%
  \let\@dtl@min=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@min}%
    {%
      \let\@dtl@min=\@dtl@num
    }%
    {%
      \dtlmin{\@dtl@min}{\@dtl@min}{\@dtl@num}%
    }%
  }%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@min}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@min}{#1}%
  }%
}
```

\DTLgminall

`\DTLgminall{⟨cmd⟩}{⟨num list⟩}`

Global version

```
\newcommand*{\DTLgminall}[2]{%
  \DTLminall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
```

\DTLmax

`\DTLmax{⟨cmd⟩}{⟨num1⟩}{⟨num2⟩}`

Sets ⟨*cmd*⟩ = max(⟨*num1*⟩, ⟨*num2*⟩)

```
\newcommand*{\DTLmax}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \DTLconverttodecimal{#3}{\@dtl@numii}%
  \dtlmax{\@dtl@tmp}{\@dtl@numi}{\@dtl@numii}%
  \dtlifnumgt{\@dtl@numi}{\@dtl@numii}%
  {%
   \dtl@ifsingle{#2}%
   {\let#1=#2}%
   {\def#1{#2}}%
  }%
  {%
   \dtl@ifsingle{#3}%
   {\let#1=#3}%
   {\def#1{#3}}%
  }%
}
```

\DTLgmax Global version

```
\newcommand*{\DTLgmax}[3]{%
  \DTLmax{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
```

\DTLmaxall

`\DTLmaxall{⟨cmd⟩}{⟨num list⟩}`

Finds the maximum value in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLmaxall}[2]{%
  \let\@dtl@max=\@empty
  \@for\dtl@thisval:=#2\do{%
```

```
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@max}%
    {%
      \let\@dtl@max\@dtl@num
    }%
    {%
      \dtlmax{\@dtl@max}{\@dtl@max}{\@dtl@num}%
    }%
  }%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@max}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@max}{#1}%
  }%
}
```

\DTLgmaxall

\DTLgmaxall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\newcommand*{\DTLgmaxall}[2]{%
\DTLmaxall{\@dtl@tmpi}{#2}%
\global\let#1=\@dtl@tmpi
}
```

\DTLmeanforall

\DTLmeanforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the arithmetic mean of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLmeanforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@mean}{#1}%
  }%
  {%
```

```
    \DTLdecimaltocurrency{\@dtl@mean}{#1}%
  }%
}
```

\DTLgmeanforall

\DTLgmeanforall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\newcommand*{\DTLgmeanforall}[2]{%
  \DTLmeanforall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
```

TLvarianceforall

\DTLvarianceforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the variance of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLvarianceforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \let\@dtl@decvals=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@decvals}%
    {%
     \let\@dtl@decvals=\@dtl@num
    }%
    {%
      \expandafter\toks@\expandafter{\@dtl@decvals}%
      \edef\@dtl@decvals{\the\toks@,\@dtl@num}%
    }%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \def\@dtl@var{0}%
  \@for\@dtl@num:=\@dtl@decvals\do{%
    \dtlsub{\@dtl@diff}{\@dtl@num}{\@dtl@mean}%
    \dtlmul{\@dtl@diff}{\@dtl@diff}{\@dtl@diff}%
    \dtladd{\@dtl@var}{\@dtl@var}{\@dtl@diff}%
  }%
  \dtldiv{\@dtl@var}{\@dtl@var}{\@dtl@n}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@var}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@var}{#1}%
  }%
}
```

Lgvarianceforall

\DTLgvarianceforall{⟨*cmd*⟩}{⟨*num list*⟩}

Global version

```
\newcommand*{\DTLgvarianceforall}[2]{%
  \DTLvarianceforall{\@dtl@tmpi}{#2}%
  \global\let#1=\@dtl@tmpi
}
```

\DTLsdforall

\DTLsdforall{⟨*cmd*⟩}{⟨*num list*⟩}

Computes the standard deviation of all the values in ⟨*num list*⟩ and stores in ⟨*cmd*⟩ which must be a control sequence.

```
\newcommand*{\DTLsdforall}[2]{%
  \def\@dtl@mean{0}%
  \def\@dtl@n{0}%
  \let\@dtl@decvals=\@empty
  \@for\dtl@thisval:=#2\do{%
    \expandafter\DTLconverttodecimal\expandafter{\dtl@thisval}{\@dtl@num}%
    \ifdefempty{\@dtl@decvals}%
    {%
     \let\@dtl@decvals=\@dtl@num
    }%
    {%
     \expandafter\toks@\expandafter{\@dtl@decvals}%
     \edef\@dtl@decvals{\the\toks@,\@dtl@num}%
    }%
    \dtladd{\@dtl@mean}{\@dtl@mean}{\@dtl@num}%
    \dtladd{\@dtl@n}{\@dtl@n}{1}%
  }%
  \dtldiv{\@dtl@mean}{\@dtl@mean}{\@dtl@n}%
  \def\@dtl@sd{0}%
  \@for\@dtl@num:=\@dtl@decvals\do{%
    \dtlsub{\@dtl@diff}{\@dtl@num}{\@dtl@mean}%
    \dtlmul{\@dtl@diff}{\@dtl@diff}{\@dtl@diff}%
    \dtladd{\@dtl@sd}{\@dtl@sd}{\@dtl@diff}%
  }%
  \dtldiv{\@dtl@sd}{\@dtl@sd}{\@dtl@n}%
  \dtlroot{\@dtl@sd}{\@dtl@sd}{2}%
```

```
    \ifdefempty{\@dtl@replaced}%
    {%
      \DTLdecimaltolocale{\@dtl@sd}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@sd}{#1}%
    }%
  }
```

\DTLgsdforall

`\DTLgsdforall{⟨cmd⟩}{⟨num list⟩}`

Global version

```
  \newcommand*{\DTLgsdforall}[2]{%
    \DTLsdforall{\@dtl@tmpi}{#2}%
    \global\let#1=\@dtl@tmpi
  }
```

\DTLround

`\DTLround{⟨cmd⟩}{⟨num⟩}{⟨num digits⟩}`

Sets ⟨*cmd*⟩ to ⟨*num*⟩ rounded to ⟨*num digits*⟩ digits after the decimal character.

```
  \newcommand*{\DTLround}[3]{%
    \DTLconverttodecimal{#2}{\@dtl@numi}%
    \dtlround{\@dtl@tmp}{\@dtl@numi}{#3}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \DTLdecimaltolocale{\@dtl@tmp}{#1}%
    }%
    {%
      \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
    }%
  }
```

\DTLground Global version

```
  \newcommand*{\DTLground}[3]{%
    \DTLround{\@dtl@tmpii}{#2}{#3}%
    \global\let#1=\@dtl@tmpii
  }
```

\DTLtrunc

`\DTLtrunc{⟨cmd⟩}{⟨num⟩}{⟨num digits⟩}`

Sets ⟨*cmd*⟩ to ⟨*num*⟩ truncated to ⟨*num digits*⟩ digits after the decimal character.

```
\newcommand*{\DTLtrunc}[3]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtltrunc{\@dtl@tmp}{\@dtl@numi}{#3}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
```

\DTLgtrunc Global version

```
\newcommand*{\DTLgtrunc}[3]{%
  \DTLtrunc{\@dtl@tmpii}{#2}{#3}%
  \global\let#1=\@dtl@tmpii
}
```

\DTLclip

```
\DTLclip{⟨cmd⟩}{⟨num⟩}
```

Sets ⟨*cmd*⟩ to ⟨*num*⟩ with all unnecessary 0's removed.

```
\newcommand*{\DTLclip}[2]{%
  \DTLconverttodecimal{#2}{\@dtl@numi}%
  \dtlclip{\@dtl@tmp}{\@dtl@numi}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \DTLdecimaltolocale{\@dtl@tmp}{#1}%
  }%
  {%
    \DTLdecimaltocurrency{\@dtl@tmp}{#1}%
  }%
}
```

\DTLgclip Global version

```
\newcommand*{\DTLgclip}[3]{%
  \DTLclip{\@dtl@tmpii}{#2}%
  \global\let#1=\@dtl@tmpii
}
```

1.5 String Macros

\DTLinitials

```
\DTLinitials{⟨string⟩}
```

Convert a string into initials. (Any ~ character found is first converted into a space.)

```
\newcommand*\DTLinitials[1]{%
  \def\dtl@initialscmd{}%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \expandafter\dtl@initials\dtl@string{} \@nil%
  \dtl@initialscmd
}%
```

The following substitutes `\protect \nobreakspace {}` with a space. (Note that in this case the space following `\nobreakspace` forms part of the command.)

```
\edef\dtl@construct@subnobrsp{%
```

Define `\@dtl@subnobrsp`

```
  \noexpand\def\noexpand\@dtl@subnobrsp##1\noexpand\protect
  \expandafter\noexpand\csname nobreakspace \endcsname ##2{%
  \noexpand\toks@{##1}%
  \noexpand\expandafter\noexpand\@dtl@toks\noexpand\expandafter{%
  \noexpand\@dtl@string}%
  \noexpand\edef\noexpand\@dtl@string{\noexpand\the\noexpand\@dtl@toks
  \noexpand\the\noexpand\toks@}%
  \noexpand\def\noexpand\@dtl@tmp{##2}%
  \noexpand\ifx\noexpand\@dtl@tmp\noexpand\@nnil
    \noexpand\let\noexpand\@dtl@subnobrspnext=\noexpand\relax
  \noexpand\else
    \noexpand\toks@{ }%
    \noexpand\expandafter\noexpand\@dtl@toks\noexpand\expandafter{%
    \noexpand\@dtl@string}%
    \noexpand\edef\noexpand\@dtl@string{\noexpand\the\noexpand\@dtl@toks
    \noexpand\the\noexpand\toks@}%
    \noexpand\let\noexpand\@dtl@subnobrspnext=\noexpand\@dtl@subnobrsp
  \noexpand\fi
  \noexpand\@dtl@subnobrspnext
  }%
```

Define `\dtl@subnobrsp`

```
  \noexpand\def\noexpand\dtl@subnobrsp##1##2{%
  \noexpand\def\noexpand\@dtl@string{}%
  \noexpand\@dtl@subnobrsp ##1\noexpand\protect\expandafter\noexpand
  \csname nobreakspace \endcsname \noexpand\@nil
  \noexpand\let##2=\noexpand\@dtl@string
  }%
}
\dtl@construct@subnobrsp
```

DTLstoreinitials

`\DTLstoreinitials{⟨string⟩}{⟨cmd⟩}`

Convert a string into initials and store in ⟨*cmd*⟩. (Any ~ character found is first converted into a space.)

```
\newcommand*{\DTLstoreinitials}[2]{%
  \def\dtl@initialscmd{}%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \expandafter\dtl@initials\dtl@string{} \@nil
  \let#2=\dtl@initialscmd
}
```

\dtl@initials

```
\def\dtl@initials#1#2 #3{%
  \dtl@ifsingle{#1}%
  {%
    \ifcat\noexpand#1\relax\relax
       \def\@dtl@donextinitials{\@dtl@initials#2 {#3}}%
    \else
       \def\@dtl@donextinitials{\@dtl@initials#1#2 {#3}}%
    \fi
  }%
  {%
    \def\@dtl@donextinitials{\@dtl@initials{#1}#2 {#3}}%
  }%
  \@dtl@donextinitials
}
```

\@dtl@initials

```
\def\@dtl@initials#1#2 #3{%
  \dtl@initialshyphen#2-{}\dtl@endhyp
  \expandafter\@dtl@toks\expandafter{\dtl@initialscmd}%
  \toks@{#1}%
  \ifdefempty{\dtl@inithyphen}%
  {%
  }%
  {%
    \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@}%
    \expandafter\@dtl@toks\expandafter{\dtl@initialscmd}%
    \expandafter\toks@\expandafter{\dtl@inithyphen}%
  }%
  \def\dtl@tmp{#3}%
  \ifx\@nnil\dtl@tmp
   \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@\DTLafterinitials}%
   \let\dtl@initialsnext=\@gobble
  \else
   \edef\dtl@initialscmd{\the\@dtl@toks\the\toks@\DTLbetweeninitials}%
   \let\dtl@initialsnext=\dtl@initials
  \fi
  \dtl@initialsnext{#3}%
}
```

@initialshyphen

```
\def\dtl@initialshyphen#1-#2#3\dtl@endhyp{%
  \def\dtl@inithyphen{#2}%
  \ifdefempty{\dtl@inithyphen}%
  {%
  }%
  {%
   \edef\dtl@inithyphen{%
     \DTLafterinitialbeforehyphen\DTLinitialhyphen#2}%
  }%
}
```

TLafterinitials

Defines what to do after the final initial.

```
\newcommand*{\DTLafterinitials}{.}
```

betweeninitials

Defines what to do between initials.

```
\newcommand*{\DTLbetweeninitials}{.}
```

ialbeforehyphen

Defines what to do before a hyphen.

```
\newcommand*{\DTLafterinitialbeforehyphen}{.}
```

TLinitialhyphen

Defines what to do at the hyphen

```
\newcommand*{\DTLinitialhyphen}{-}
```

TLifAllUpperCase

`\DTLifAllUpperCase{⟨string⟩}{⟨true part⟩}{⟨false part⟩}`

If ⟨*string*⟩ only contains uppercase characters do ⟨*true part*⟩, otherwise do ⟨*false part*⟩.

```
\newcommand*{\DTLifAllUpperCase}[3]{%
  \protected@edef\dtl@tuc{#1}%
  \expandafter\dtl@testifuppercase\dtl@tuc\@nil\relax
  \if@dtl@condition#2\else#3\fi
}
```

tifalluppercase

```
\def\dtl@testifuppercase#1#2{%
  \def\dtl@argi{#1}%
  \def\dtl@argii{#2}%
  \def\dtl@tc@rest{}%
  \ifx\dtl@argi\@nnil
    \let\dtl@testifuppernext=\@nnil
  \else
    \ifx#1\protect
```

```
      \let\dtl@testifuppernext=\dtl@testifuppercase
    \else
      \ifx\uppercase#1\relax
        \@dtl@conditiontrue
        \def\dtl@tc@rest{}%
        \let\dtl@testifuppernext=\relax
      \else
        \edef\dtl@tc@arg{\string#1}%
        \expandafter\dtl@test@ifuppercase\dtl@tc@arg\end
        \ifx\dtl@argii\@nnil
          \let\dtl@testifuppernext=\@dtl@gobbletonil
        \fi
      \fi
    \fi
  \fi
  \ifx\dtl@testifuppernext\relax
   \edef\dtl@dotestifuppernext{%
     \noexpand\dtl@testifuppercase}%
  \else
   \ifx\dtl@testifuppernext\@nnil
     \edef\dtl@dotestifuppernext{#2}%
   \else
     \expandafter\toks@\expandafter{\dtl@tc@rest}%
     \@dtl@toks{#2}%
     \edef\dtl@dotestifuppernext{%
       \noexpand\dtl@testifuppernext\the\toks@\the\@dtl@toks}%
   \fi
  \fi
  \dtl@dotestifuppernext
}
```

@ifalluppercase

```
\def\dtl@test@ifuppercase#1#2\end{%
  \def\dtl@tc@rest{#2}%
  \IfSubStringInString{\string\MakeUppercase}{#1#2}%
  {%
     \@dtl@conditiontrue
     \def\dtl@tc@rest{}%
     \let\dtl@testifuppernext=\relax
  }%
  {%
    \IfSubStringInString{\string\MakeTextUppercase}{#1#2}%
    {%
       \@dtl@conditiontrue
       \def\dtl@tc@rest{}%
       \let\dtl@testifuppernext=\relax
    }%
    {%
      \edef\dtl@uccode{\the\uccode`#1}%
      \edef\dtl@code{\number`#1}%
      \ifnum\dtl@code=\dtl@uccode\relax
        \@dtl@conditiontrue
        \let\dtl@testifuppernext=\dtl@testifuppercase
      \else
        \ifnum\dtl@uccode=0\relax
          \@dtl@conditiontrue
          \let\dtl@testifuppernext=\dtl@testifuppercase
        \else
          \@dtl@conditionfalse
          \let\dtl@testifuppernext=\@dtl@gobbletonil
        \fi
      \fi
    }%
  }%
}
```

TLifAllLowerCase

\DTLifAllLowerCase{⟨*string*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If ⟨*string*⟩ only contains lowercase characters do ⟨*true part*⟩, otherwise do ⟨*false part*⟩.

```
\newcommand*{\DTLifAllLowerCase}[3]{%
  \protected@edef\dtl@tlc{#1}%
  \expandafter\dtl@testiflowercase\dtl@tlc\@nil\relax
  \if@dtl@condition#2\else#3\fi
}
```

tifalllowercase

```
\def\dtl@testiflowercase#1#2{%
  \def\dtl@argi{#1}%
  \def\dtl@argii{#2}%
  \ifx\dtl@argi\@nnil
    \let\dtl@testiflowernext=\@nnil
  \else
    \ifx#1\protect
      \let\dtl@testiflowernext=\dtl@testiflowercase
    \else
      \ifx\lowercase#1\relax
        \@dtl@conditiontrue
        \def\dtl@tc@rest{}%
        \let\dtl@testiflowernext=\relax
      \else
        \edef\dtl@tc@arg{\string#1}%
        \expandafter\dtl@test@iflowercase\dtl@tc@arg\end
        \ifx\dtl@argii\@nnil
          \let\dtl@testiflowernext=\@dtl@gobbletonil
        \fi
      \fi
    \fi
  \fi
  \ifx\dtl@testiflowernext\relax
   \edef\dtl@dotestiflowernext{%
     \noexpand\dtl@testiflowercase}%
  \else
   \ifx\dtl@testiflowernext\@nnil
     \edef\dtl@dotestiflowernext{#2}%
   \else
     \expandafter\toks@\expandafter{\dtl@tc@rest}%
     \@dtl@toks{#2}%
     \edef\dtl@dotestiflowernext{%
       \noexpand\dtl@testiflowernext\the\toks@\the\@dtl@toks}%
   \fi
  \fi
  \dtl@dotestiflowernext
}
```

@ifalllowercase

```
\def\dtl@test@iflowercase#1#2\end{%
  \def\dtl@tc@rest{#2}%
  \IfSubStringInString{\string\MakeLowercase}{#1#2}%
  {%
    \@dtl@conditiontrue
    \def\dtl@tc@rest{}%
    \let\dtl@testiflowernext=\relax
  }%
  {%
    \IfSubStringInString{\string\MakeTextLowercase}{#1#2}%
    {%
      \@dtl@conditiontrue
      \def\dtl@tc@rest{}%
      \let\dtl@testiflowernext=\relax
    }%
    {%
      \edef\dtl@lccode{\the\lccode`#1}%
      \edef\dtl@code{\number`#1}%
      \ifnum\dtl@code=\dtl@lccode\relax
        \@dtl@conditiontrue
        \let\dtl@testiflowernext=\dtl@testiflowercase
      \else
        \ifnum\dtl@lccode=0\relax
          \@dtl@conditiontrue
          \let\dtl@testiflowernext=\dtl@testiflowercase
        \else
          \@dtl@conditionfalse
          \let\dtl@testiflowernext=\@dtl@gobbletonil
        \fi
      \fi
    }%
  }%
}
```

\DTLsubstitute

```
\DTLsubstitute{⟨cmd⟩}{⟨original⟩}{⟨replacement⟩}
```

Substitutes first occurrence of ⟨*original*⟩ with {⟨*replacement*⟩} within the string given by ⟨*cmd*⟩

```
\newcommand{\DTLsubstitute}[3]{%
  \expandafter\DTLsplitstring\expandafter
    {#1}{#2}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
  {%
  }%
  {%
    \def#1{}%
    \expandafter\@dtl@toks\expandafter{\@dtl@beforepart}%
    \expandafter\toks@\expandafter{#1}%
    \protected@edef#1{\the\toks@\the\@dtl@toks#3}%
    \expandafter\@dtl@toks\expandafter{\@dtl@afterpart}%
    \expandafter\toks@\expandafter{#1}%
    \edef#1{\the\toks@\the\@dtl@toks}%
  }%
}
```

\DTLsplitstring

```
\DTLsplitstring{⟨string⟩}{⟨split text⟩}{⟨before cmd⟩}{⟨after cmd⟩}
```

Splits string at ⟨*split text*⟩ stores the pre split text in ⟨*before cmd*⟩ and the post split text in ⟨*after cmd*⟩.

```
\newcommand*{\DTLsplitstring}[4]{%
  \def\dtl@splitstr##1#2##2\@nil{%
    \def#3{##1}%
    \def#4{##2}%
    \ifdefempty{#4}%
    {%
      \let\@dtl@replaced=\@empty
    }%
    {%
      \def\@dtl@replaced{#2}%
      \dtl@split@str##2\@nil
    }%
  }%
  \def\dtl@split@str##1#2\@nil{\def#4{##1}}%
  \dtl@splitstr#1#2\@nil
}
```

DTLsubstituteall

```
\DTLsubstituteall{⟨cmd⟩}{⟨original⟩}{⟨replacement⟩}
```

Substitutes all occurrences of ⟨*original*⟩ with {⟨*replacement*⟩} within the string given by ⟨*cmd*⟩

```
\newcommand{\DTLsubstituteall}[3]{%
  \def\@dtl@splitsubstr{}%
  \let\@dtl@afterpart=#1\relax
  \@dtl@dosubstitute{#2}{#3}%
  \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
  \expandafter\@dtl@toks\expandafter{\@dtl@afterpart}%
  \long\edef#1{\the\toks@\the\@dtl@toks}%
}
```

tl@dosubstitute Recursive substitution macro.

```
\def\@dtl@dosubstitute#1#2{%
  \expandafter\DTLsplitstring\expandafter
   {\@dtl@afterpart}{#1}{\@dtl@beforepart}{\@dtl@afterpart}%
  \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
  \expandafter\@dtl@toks\expandafter{\@dtl@beforepart}%
  \edef\@dtl@splitsubstr{\the\toks@\the\@dtl@toks}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \let\@dtl@dosubstnext=\@dtl@dosubstitutenoop
  }%
  {%
    \expandafter\toks@\expandafter{\@dtl@splitsubstr}%
    \@dtl@toks{#2}%
    \edef\@dtl@splitsubstr{\the\toks@\the\@dtl@toks}%
    \let\@dtl@dosubstnext=\@dtl@dosubstitute
  }%
  \@dtl@dosubstnext{#1}{#2}%
}
```

osubstitutenoop Terminates recursive substitution macro.

```
\def\@dtl@dosubstitutenoop#1#2{}
```

1.6 Conditionals

f@dtl@condition

```
\newif\if@dtl@condition
```

1.6.1 Determining Data Types

The control sequence \@dtl@checknumerical checks the data type of its argument, and sets \@dtl@datatype to 0 if the argument is a string, 1 if the argument is an integer or 2 if the argument is a real number. First define \@dtl@datatype:

\@dtl@datatype

```
\newcount\@dtl@datatype
```

l@checknumerical

\@dtl@checknumerical{⟨*arg*⟩}

Checks if ⟨*arg*⟩ is numerical (includes decimal numbers, but not scientific notation.) Sets \@dtl@datatype, as described above.

```
\newcommand{\@dtl@checknumerical}[1]{%
  \@dtl@numgrpsepfalse
  \dtl@ifsingle{#1}%
  {%
    \expandafter\toks@\expandafter{#1}%
    \edef\@dtl@tmp{\the\toks@}%
  }%
  {%
    \def\@dtl@tmp{#1}%
  }%

  \ifdefempty\@dtl@tmp
  {%
    \@dtl@datatype=0\relax
  }%
  {%
    \@dtl@tmpcount=0\relax
    \@dtl@datatype=0\relax
    \@dtl@numgrpsepcount=2\relax
    \@dtl@standardize@currency\@dtl@tmp
    \ifdefempty{\@dtl@org@currency}%
    {%
    }%
    {%
      \let\@dtl@currency\@dtl@org@currency
    }%
    \expandafter\@dtl@checknumericalstart\@dtl@tmp\@nil\@nil
  }%
  \ifnum\@dtl@numgrpsepcount>-1\relax
    \if@dtl@numgrpsep
      \ifnum\@dtl@numgrpsepcount=3\relax
      \else
        \@dtl@datatype=0\relax
      \fi
    \fi
  \fi
}
```

\@dtl@protect

```
\newcommand*{\@dtl@protect}{\protect}
```

\@dtl@minus

```
\newcommand*{\@dtl@minus}{-}
```

\@dtl@plus

```
\newcommand*{\@dtl@plus}{+}
```

\@dtl@dollar

```
\newcommand*{\@dtl@dollar}{\$}
```

knumericalstart Check first character for checknumerical process to see if it's a plus or minus sign.

```
\def\@dtl@checknumericalstart#1#2\@nil\@nil{%
  \def\@dtl@tmp{#1}%
  \ifx\@dtl@tmp\@dtl@protect
    \@dtl@checknumericalstart#2\@nil\@nil\relax
  \else
    \ifx\@dtl@tmp\@dtl@minus
      \def\@dtl@tmp{#2}%
      \ifdefempty{\@dtl@tmp}%
      {%
        \@dtl@datatype=0\relax
      }%
      {%
        \ifnum\@dtl@datatype=0\relax
          \@dtl@datatype=1\relax
        \fi
        \@dtl@checknumericalstart#2\@nil\@nil\relax
      }%
    \else
      \ifx\@dtl@tmp\@dtl@plus
        \def\@dtl@tmp{#2}%
        \ifdefempty{\@dtl@tmp}%
        {%
          \@dtl@datatype=0\relax
        }%
        {%
          \ifnum\@dtl@datatype=0\relax
            \@dtl@datatype=1\relax
          \fi
          \@dtl@checknumericalstart#2\@nil\@nil\relax
        }%
      \else
        \def\@dtl@tmp{#1}%
        \ifx\@dtl@tmp\@dtl@dollar
          \def\@dtl@tmp{#2}%
          \ifdefempty{\@dtl@tmp}%
          {%
            \@dtl@datatype=0\relax
          }%
          {%
            \@dtl@datatype=3\relax
            \@dtl@checknumericalstart#2\@nil\@nil\relax
          }%
        \else
          \ifdefempty{\@dtl@tmp}%
          {%
            \@dtl@datatype=0\relax
          }%
          {%
            \ifnum\@dtl@datatype=0\relax
              \@dtl@datatype=1\relax
            \fi
            \@dtl@checknumericalloop#1#2\@nil\@nil\relax
          }%
        \fi
      \fi
    \fi
  \fi
}
```

f@dtl@numgrpsep The conditional `\if@dtl@numgrpsep` is set the first time `\@dtl@checknumericalloop` encounters the number group separator.

```
\newif\if@dtl@numgrpsep
```

gitOrDecimalSep Check if argument is either a digit or the decimal separator. Rewrite provided by Bruno Le Floch.

```
\newcommand*{\@dtl@ifDigitOrDecimalSep}[3]{%
  \ifnum 9<1\noexpand#1\relax
    #2%
  \else
    \expandafter\ifx\@dtl@decimal#1\relax
      #2%
    \else
      #3%
    \fi
  \fi
}
```

cknumericalloop Check numerical loop. This iterates through each character until `\@nil` is reached, or invalid character found. Increments `\@dtl@tmpcount` each time it encounters a decimal character.

```
\def\@dtl@checknumericalloop#1#2\@nil{%
\def\@dtl@tmp{#1}%
\ifx\@nnil\@dtl@tmp\relax
 \let\@dtl@chcknumnext=\@dtl@checknumericalnoop%
\else
  \@dtl@ifDigitOrDecimalSep{#1}{%
  \let\@dtl@chcknumnext=\@dtl@checknumericalloop%
  \expandafter\ifx\@dtl@decimal#1\relax
     \if@dtl@numgrpsep
       \ifnum\@dtl@numgrpsepcount=3\relax
        \@dtl@numgrpsepcount=-1\relax
       \else
        \@dtl@datatype=0\relax
        \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
       \fi
     \else
       \@dtl@numgrpsepcount=-1\relax
     \fi
   \else
     \ifnum\@dtl@numgrpsepcount=-1\relax
     \else
       \advance\@dtl@numgrpsepcount by 1\relax
     \fi
   \fi
}{%
\ifx\@dtl@numbergroupchar\@dtl@tmp\relax
  \@dtl@numgrpseptrue
  \ifnum\@dtl@numgrpsepcount<3\relax
    \@dtl@datatype=0\relax
    \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
  \else
     \@dtl@numgrpsepcount=0\relax
  \fi
\else
  \@dtl@datatype=0\relax
  \let\@dtl@chcknumnext=\@dtl@checknumericalnoop
\fi
}%
  \ifx\@dtl@decimal\@dtl@tmp\relax
   \ifnum\@dtl@datatype<3\relax
     \@dtl@datatype=2\relax
   \fi
   \advance\@dtl@tmpcount by 1\relax
   \ifnum\@dtl@tmpcount>1\relax
     \@dtl@datatype=0\relax
     \let\@dtl@chcknumnext=\@dtl@checknumericalnoop%
   \fi
  \fi
\fi
\@dtl@chcknumnext#2\@nil
}
```

cknumericalnoop End loop

```
\def\@dtl@checknumericalnoop#1\@nil#2{}
```

\DTLifnumerical

```
\DTLifnumerical{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

Tests the first argument, if its numerical do second argument, otherwise do third argument.

```
\newcommand{\DTLifnumerical}[3]{%
\@dtl@checknumerical{#1}%
\ifnum\@dtl@datatype=0\relax#3\else#2\fi
}
```

\DTLifreal

```
\DTLifreal{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

Tests the first argument, if it's a real number (not an integer) do second argument, otherwise do third argument.

```
\newcommand{\DTLifreal}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=2\relax #2\else #3\fi
}
```

\DTLifint

```
\DTLifint{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

Tests the first argument, if it's an integer do second argument, otherwise do third argument.

```
\newcommand{\DTLifint}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=1\relax #2\else #3\fi
}
```

\DTLifstring

```
\DTLifstring{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

Tests the first argument, if it's a string do second argument, otherwise do third argument.

```
\newcommand{\DTLifstring}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=0\relax #2\else #3\fi
}
```

\DTLifcurrency

```
\DTLifcurrency{⟨arg⟩}{⟨true part⟩}{⟨false part⟩}
```

Tests the first argument, if it starts with the currency symbol do second argument, otherwise do third argument.

```
\newcommand{\DTLifcurrency}[3]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=3\relax #2\else #3\fi
}
```

TLifcurrencyunit

```
\DTLifcurrencyunit{⟨arg⟩}{⟨symbol⟩}{⟨true part⟩}{⟨false part⟩}
```

This tests if ⟨*arg*⟩ is currency, and uses the currency unit ⟨*symbol*⟩. If true do third argument, otherwise do fourth argument.

```
\newcommand*{\DTLifcurrencyunit}[4]{%
  \@dtl@checknumerical{#1}%
  \ifnum\@dtl@datatype=3\relax
    \ifthenelse{\equal{\@dtl@org@currency}{#2}}{#3}{#4}%
  \else
    #4%
  \fi
}
```

TLifcasedatatype

```
\DTLifcasedatatype{⟨arg⟩}{⟨string case⟩}{⟨int case⟩}{⟨real case⟩}
{⟨currency case⟩}
```

If ⟨*arg*⟩ is a string, do ⟨*string case*⟩, if ⟨*arg*⟩ is an integer do ⟨*int case*⟩, if ⟨*arg*⟩ is a real number, do ⟨*real case*⟩, if ⟨*arg*⟩ is currency, do ⟨*curreny case*⟩.

```
\newcommand{\DTLifcasedatatype}[5]{%
  \@dtl@checknumerical{#1}%
  \ifcase\@dtl@datatype
   #2% string
  \or
   #3% integer
  \or
   #4% number
  \or
   #5% currency
  \fi
}
```

estbothnumerical

\dtl@testbothnumerical{⟨*arg1*⟩}{⟨*arg2*⟩}

Tests if both arguments are numerical. This sets the conditional \if@dtl@condition.

```
\newcommand*{\dtl@testbothnumerical}[2]{%
  \dtl@ifsingle{#1}{%
   \edef\@dtl@tmp{#1}}{%
   \def\@dtl@tmp{#1}}%
  \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
  \edef\@dtl@firsttype{\number\@dtl@datatype}%
  \dtl@ifsingle{#2}{%
   \edef\@dtl@tmp{#2}}{%
   \def\@dtl@tmp{#2}}%
  \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
  \multiply\@dtl@datatype by \@dtl@firsttype\relax
  \ifnum\@dtl@datatype>0\relax
   \@dtl@conditiontrue
  \else
   \@dtl@conditionfalse
  \fi
}
```

\DTLifnumlt

\DTLifnumlt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if {⟨*num1*⟩} < {⟨*num2*⟩}. Both numbers need to have the decimal separator changed to a dot to ensure that it works with \dtlifnumlt

```
\newcommand*{\DTLifnumlt}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumlt{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
```

tlcompareskipcs

If true, \dtlcompare should skip control sequences.

```
\newif\ifdtlcompareskipcs
\dtlcompareskipcsfalse
```

\dtlcompare

\dtlcompare{⟨*count*⟩}{⟨*string1*⟩}{⟨*string2*⟩}

Compares ⟨*string1*⟩ and ⟨*string2*⟩, and stores the result in the count register ⟨*count*⟩. The result may be one of:

-1 if ⟨*string1*⟩ is considered to be less than ⟨*string2*⟩
0 if ⟨*string1*⟩ is considered to be the same as ⟨*string2*⟩
1 if ⟨*string1*⟩ is considered to be greater than ⟨*string2*⟩

Note that for the purposes of string comparisons, commands within ⟨*string1*⟩ and ⟨*string2*⟩ are ignored, except for `\space` and `~`, which are both treated as a space (character code 32.) The following examples assume that the count register `\mycount` has been defined as follows:

```
\newcount\mycount
```

Examples:

1. `\dtlcompare{\mycount}{Z\"oe}{Zoe}\number\mycount`

   produces: -1, since the accent command is ignored.

2. `\dtlcompare{\mycount}{foo}{Foo}\number\mycount`

   produces: 1, since the comparison is case sensitive, however, note the following example:

3. `\dtlcompare{\mycount}{foo}{\uppercase{f}oo}\number\mycount`

   which produces: 1, since the `\uppercase` command is ignored.

4. A control sequence is treated as having the character code value of 0. Pre version 2.32, `\dtlcompare` was advertised here as skipping control sequences when actually it was treating a control sequence as character 0. To avoid breaking backward-compatibility where a control sequence is expected to have this behaviour, we now have a switch to determine whether to treat control sequences as 0 or to skip them.

   So you can now "trick" `\dtlcompare` using a command which doesn't output any text if switch this conditional on. Suppose you have defined the following command:

   ```
   \newcommand*{\noopsort}[1]{}
   ```

   then `\noopsort{a}foo` produces the text: foo, however the following

   ```
   \dtlcompare{\mycount}{\noopsort{a}foo}{bar}\number\mycount
   ```

   produces: -1, since the command `\noopsort` is disregarded when the comparison is made, so `\dtlcompare` just compares `{a}foo` with `bar`, and since `a` is less than `b`, the first string is considered to be less than the second string.

5. Note that this also means that:

   ```
   \def\mystr{abc}%
   \dtlcompare{\mycount}{\mystr}{abc}\number\mycount
   ```

produces: -1, since the command \mystr is disregarded, which means that \dtlcompare is comparing an empty string with the string abc.

6. Spaces count in the comparison:

   ```
   \dtlcompare{\mycount}{ab cd}{abcd}\number\mycount
   ```

   produces: -1, but sequential spaces are treated as a single space:

   ```
   \dtlcompare{\mycount}{ab  cd}{ab cd}\number\mycount
   ```

   produces: 0.

7. As usual, spaces following command names are ignored, so

   ```
   \dtlcompare{\mycount}{ab\relax cd}{ab cd}\number\mycount
   ```

   produces: -1.

8. ~ and \space are considered to be the same as a space:

   ```
   \dtlcompare{\mycount}{ab cd}{ab~cd}\number\mycount
   ```

   produces: 0.

```
\newcommand*{\dtlcompare}[3]{%
  \dtl@subnobrsp{#2}{\@dtl@argA}%
  \dtl@subnobrsp{#3}{\@dtl@argB}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=1\relax
    }%
    {%
```

Identify all word breaks.

```
      \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
      \let\@dtl@argA\dtl@string
      \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
```

```
\let\@dtl@argB\dtl@string
\expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
\let\dtl@firstA=\dtl@first
\let\dtl@restA=\dtl@rest
\expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
\let\dtl@firstB=\dtl@first
\let\dtl@restB=\dtl@rest
\expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstA}%
{%
  \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstB}%
  {%
    \expandafter\dtl@setcharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
    \expandafter\dtl@setcharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
```

If control sequences should be skipped, check if either is a control sequence (the character code will be 0).

```
    \let\dtl@donextcompare\@firstofone
    \ifdtlcompareskipcs
      \ifnum\dtl@codeA=0\relax
        \ifnum\dtl@codeB=0\relax
```

Skip both.

```
          \edef\dtl@donext{%
            \noexpand\dtlcompare
            {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \else
```

Skip A.

```
          \edef\dtl@donext{%
            \noexpand\dtlcompare
            {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\@dtl@argB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \fi
      \else
        \ifnum\dtl@codeB=0\relax
```

Skip B.

```
          \edef\dtl@donext{%
            \noexpand\dtlcompare
            {\noexpand#1}{\expandonce\@dtl@argA}{\expandonce\dtl@restB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \fi
      \fi
    \fi
    \dtl@donextcompare
    {%
      \ifnum\dtl@codeA=-1\relax
```

```
  \ifnum\dtl@codeB=-1\relax
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
     \edef\dtl@donext{%
       \noexpand\dtlcompare
       {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
     \dtl@donext
  \else
     \edef\dtl@donext{%
       \noexpand\dtlcompare
         {\noexpand#1}%
         {\expandonce\dtl@restA}%
         {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
     \dtl@donext
  \fi
\else
  \ifnum\dtl@codeB=-1\relax
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
     \edef\dtl@donext{%
       \noexpand\dtlcompare
         {\noexpand#1}%
         {\expandonce\dtl@firstA\expandonce\dtl@restA}%
         {\expandonce\dtl@restB}}%
     \dtl@donext
  \else
    \ifnum\dtl@codeA<\dtl@codeB
      #1=-1\relax
    \else
      \ifnum\dtl@codeA>\dtl@codeB
        #1=1\relax
      \else
         \ifdefempty{\dtl@restA}%
         {%
           \ifdefempty{\dtl@restB}%
           {%
             #1=0\relax
           }%
           {%
             #1=-1\relax
           }%
         }%
         {%
           \ifdefempty{\dtl@restB}%
           {%
             #1=1\relax
           }%
           {%
             \protected@edef\dtl@donext{%
               \noexpand\dtlcompare
```

```
                             {\noexpand#1}{\dtl@restA}{\dtl@restB}}%
                         \dtl@donext
                       }%
                     }%
                  \fi
                \fi
              \fi
            \fi
          }%
        }%
        {%
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
          \edef\dtl@donext{%
            \noexpand\dtlcompare
              {\noexpand#1}%
              {\expandonce\dtl@firstA\expandonce\dtl@restA}%
              {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
          \dtl@donext
        }%
      }%
      {%
        \edef\dtl@donext{%
          \noexpand\dtlcompare
            {\noexpand#1}%
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
        \dtl@donext
      }%
    }%
  }%
}
```

l@if@two@octets

Check if argument starts with \UTFviii@two@octets

```
\def\dtl@if@two@octets#1#2\dtl@end@if@two@octets#3#4{%
  \ifbool{@dtl@utf8}
  {%
    \ifx\UTFviii@two@octets#1\relax
     #3%
    \else
     #4%
    \fi
  }%
  {%
    #4%
  }%
}
```

etfirst@UTFviii

```
\def\dtl@getfirst@UTFviii#1#2#3\end@dtl@getfirst@UTFviii{%
  \def\dtl@first{#1#2}%
  \ifx\@nil#3\relax
    \def\dtl@rest{}%
  \else
    \expandafter\def\expandafter\dtl@rest\expandafter{\@dtl@firsttonil#3}%
  \fi
}
```

@dtl@firsttonil

```
\def\@dtl@firsttonil#1\@nil{#1}
```

\dtl@getfirst

Gets the first object, and stores in `\dtl@first`. The remainder is stored in `\dtl@rest`.

```
\def\dtl@getfirst#1#2\end@dtl@getfirst{%
  \def\dtl@first{#1}%
  \ifdefempty{\dtl@first}%
  {%
    \def\dtl@rest{#2}%
  }%
  {%
    \ifbool{@dtl@utf8}
    {%
      \expandafter\dtl@if@two@octets#1#2\relax\dtl@end@if@two@octets
      {%
        \dtl@getfirst@UTFviii#1#2\@nil\end@dtl@getfirst@UTFviii
      }%
      {%
        \dtl@ifsingle{#1}{\def\dtl@rest{#2}}{\dtl@getfirst#1#2\end@dtl@getfirst}%
      }%
    }%
    {%
      \dtl@ifsingle{#1}{\def\dtl@rest{#2}}{\dtl@getfirst#1#2\end@dtl@getfirst}%
    }%
  }%
}%
```

Count registers to store character codes:

```
\newcount\dtl@codeA
\newcount\dtl@codeB
```

\dtl@setcharcode

`\dtl@setcharcode{⟨c⟩}{⟨count register⟩}`

Sets ⟨*count register*⟩ to the character code of ⟨*c*⟩, or to −1 if ⟨*c*⟩ is empty, or to 0 if ⟨*c*⟩ is a control sequence, unless ⟨*c*⟩ is either `\space` or ˜ in which case it sets ⟨*count register*⟩ to the character code of the space character.

```
\newcommand*{\dtl@setcharcode}[2]{%
```

```
  \ifstrempty{#1}%
  {%
```

Empty argument. Set code to −1.

```
    #2=-1\relax
  }%
  {%
    \ifx\@dtl@wordbreak#1\relax
```

Reached a word break. Set to character code of a space.

```
     #2=`\ \relax
   \else
     \ifcat\noexpand#1\relax
```

Argument is a control sequence, so set to 0.

```
       #2=0\relax
     \else
       \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
       {%
```

Argument is a UTF8 character.

```
        \dtlsetUTFviiicharcode{#1}{#2}%
       }%
       {%
```

Argument is a character, so set to the character code.

```
         \dtlsetcharcode{#1}{#2}%
       }%
     \fi
   \fi
  }%
}
```

\dtlsetcharcode Set the code for the given character. May be redefined by user for non-UTF8 encodings (e.g. Latin-1).

```
\newcommand*{\dtlsetcharcode}[2]{#2=`#1\relax}
```

tlsetlccharcode Set the lowercase code for the given character. May be redefined by user for non-UTF8 encodings (e.g. Latin-1).

```
\newcommand*{\dtlsetlccharcode}[2]{#2=\lccode`#1\relax}
```

UTFviiicharcode Default behaviour is to set all UTF8 characters to code 64 (before A). This will need to be redefined according to the relevant alphabet.

```
\newcommand*\dtlsetUTFviiicharcode[2]{\dtlsetdefaultUTFviiicharcode{#1}{#2}}
```

Fviiilccharcode Default behaviour is to set all UTF8 characters to code 96 (before a). This will need to be redefined according to the relevant alphabet.

```
\newcommand*\dtlsetUTFviiilccharcode[2]{\dtlsetdefaultUTFviiilccharcode{#1}{#2}}
```

UTFviiicharcode Default codes for some supplemental Latin characters.

```
\newcommand*\dtlsetdefaultUTFviiicharcode[2]{%
 \ifboolexpr
 {
      test {\ifstrequal{#1}{À}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Ã}}
   or test {\ifstrequal{#1}{Ä}}
 }%
 {%
   #2=`A\relax
 }%
 {%
   \ifstrequal{#1}{Ç}%
   {%
     #2=`C\relax
   }%
   {%
     \ifboolexpr
     {
          test {\ifstrequal{#1}{È}}
       or test {\ifstrequal{#1}{É}}
       or test {\ifstrequal{#1}{Ê}}
       or test {\ifstrequal{#1}{Ë}}
     }%
     {%
       #2=`E\relax
     }%
     {%
       \ifboolexpr
       {
            test {\ifstrequal{#1}{Ì}}
         or test {\ifstrequal{#1}{Í}}
         or test {\ifstrequal{#1}{Î}}
         or test {\ifstrequal{#1}{Ï}}
       }%
       {%
         #2=`I\relax
       }%
       {%
         \ifstrequal{#1}{Ñ}%
         {%
           #2=`N\relax
         }%
         {%
           \ifboolexpr
           {
                test {\ifstrequal{#1}{Ò}}
```

```
  or test {\ifstrequal{#1}{Ó}}
  or test {\ifstrequal{#1}{Ô}}
  or test {\ifstrequal{#1}{Õ}}
  or test {\ifstrequal{#1}{Ö}}
}%
{%
  #2=‘O\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{Ù}}
    or test {\ifstrequal{#1}{Ú}}
    or test {\ifstrequal{#1}{Û}}
    or test {\ifstrequal{#1}{Ü}}
  }%
  {%
    #2=‘U\relax
  }%
  {%
    \ifstrequal{#1}{Ý}%
    {%
      #2=‘Y\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{à}}
        or test {\ifstrequal{#1}{á}}
        or test {\ifstrequal{#1}{á}}
        or test {\ifstrequal{#1}{ã}}
        or test {\ifstrequal{#1}{ä}}
      }%
      {%
        #2=‘a\relax
      }%
      {%
        \ifstrequal{#1}{ç}%
        {%
          #2=‘c\relax
        }%
        {%
          \ifboolexpr
          {
               test {\ifstrequal{#1}{è}}
            or test {\ifstrequal{#1}{é}}
            or test {\ifstrequal{#1}{ê}}
            or test {\ifstrequal{#1}{ë}}
          }%
```

```
{%
  #2=`e\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{ì}}
    or test {\ifstrequal{#1}{í}}
    or test {\ifstrequal{#1}{î}}
    or test {\ifstrequal{#1}{ï}}
  }%
  {%
    #2=`i\relax
  }%
  {%
    \ifstrequal{#1}{ñ}%
    {%
      #2=`n\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ò}}
        or test {\ifstrequal{#1}{ó}}
        or test {\ifstrequal{#1}{ô}}
        or test {\ifstrequal{#1}{õ}}
        or test {\ifstrequal{#1}{ö}}
      }%
      {%
        #2=`o\relax
      }%
      {%
        \ifboolexpr
        {
             test {\ifstrequal{#1}{ù}}
          or test {\ifstrequal{#1}{ú}}
          or test {\ifstrequal{#1}{û}}
          or test {\ifstrequal{#1}{ü}}
        }%
        {%
          #2=`u\relax
        }%
        {%
          \ifstrequal{#1}{ý}%
          {%
            #2=`y\relax
          }%
          {%
            #2=64\relax
```

```
                               }%
                             }%
                           }%
                         }%
                       }%
                     }%
                   }%
                 }%
               }%
             }%
           }%
         }%
       }%
     }%
   }%
 }%
}
```

Fviiilccharcode As above but for case-insensitive comparison.

```
\newcommand*\dtlsetdefaultUTFviiilccharcode[2]{%
 \ifboolexpr
 {
      test {\ifstrequal{#1}{à}}
   or test {\ifstrequal{#1}{á}}
   or test {\ifstrequal{#1}{á}}
   or test {\ifstrequal{#1}{ã}}
   or test {\ifstrequal{#1}{ä}}
   or test {\ifstrequal{#1}{À}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Á}}
   or test {\ifstrequal{#1}{Ã}}
   or test {\ifstrequal{#1}{Ä}}
 }%
 {%
   #2=‘a\relax
 }%
 {%
   \ifboolexpr
   {
        test {\ifstrequal{#1}{ç}}
     or test {\ifstrequal{#1}{Ç}}
   }
   {%
     #2=‘c\relax
   }%
   {%
     \ifboolexpr
     {
          test {\ifstrequal{#1}{è}}
```

```
  or test {\ifstrequal{#1}{é}}
  or test {\ifstrequal{#1}{ê}}
  or test {\ifstrequal{#1}{ë}}
  or test {\ifstrequal{#1}{È}}
  or test {\ifstrequal{#1}{É}}
  or test {\ifstrequal{#1}{Ê}}
  or test {\ifstrequal{#1}{Ë}}
}%
{%
  #2=`e\relax
}%
{%
  \ifboolexpr
  {
       test {\ifstrequal{#1}{ì}}
    or test {\ifstrequal{#1}{í}}
    or test {\ifstrequal{#1}{î}}
    or test {\ifstrequal{#1}{ï}}
    or test {\ifstrequal{#1}{Ì}}
    or test {\ifstrequal{#1}{Í}}
    or test {\ifstrequal{#1}{Î}}
    or test {\ifstrequal{#1}{Ï}}
  }%
  {%
    #2=`i\relax
  }%
  {%
    \ifboolexpr
    {
         test {\ifstrequal{#1}{ñ}}
      or test {\ifstrequal{#1}{Ñ}}
    }
    {%
      #2=`n\relax
    }%
    {%
      \ifboolexpr
      {
           test {\ifstrequal{#1}{ò}}
        or test {\ifstrequal{#1}{ó}}
        or test {\ifstrequal{#1}{ô}}
        or test {\ifstrequal{#1}{õ}}
        or test {\ifstrequal{#1}{ö}}
        or test {\ifstrequal{#1}{Ò}}
        or test {\ifstrequal{#1}{Ó}}
        or test {\ifstrequal{#1}{Ô}}
        or test {\ifstrequal{#1}{Õ}}
        or test {\ifstrequal{#1}{Ö}}
      }%
```

```
           {%
             #2=‘o\relax
           }%
           {%
             \ifboolexpr
             {
                  test {\ifstrequal{#1}{ù}}
               or test {\ifstrequal{#1}{ú}}
               or test {\ifstrequal{#1}{û}}
               or test {\ifstrequal{#1}{ü}}
               or test {\ifstrequal{#1}{Ù}}
               or test {\ifstrequal{#1}{Ú}}
               or test {\ifstrequal{#1}{Û}}
               or test {\ifstrequal{#1}{Ü}}
             }%
             {%
               #2=‘u\relax
             }%
             {%
               \ifboolexpr
               {
                    test {\ifstrequal{#1}{ý}}
                 or test {\ifstrequal{#1}{Ý}}
               }%
               {%
                 #2=‘y\relax
               }%
               {%
                 #2=96\relax
               }%
             }%
           }%
         }%
       }%
     }%
   }%
 }%
}
```

tl@setlccharcode

`\dtl@setlccharcode{⟨c⟩}{⟨count register⟩}`

As `\dtl@setlccharcode` except it sets ⟨*count register*⟩ to the lower case character code of ⟨*c*⟩, unless ⟨*c*⟩ is a control sequence, in which case it does the same as `\dtl@setcharcode`.

```
\newcommand*{\dtl@setlccharcode}[2]{%
  \ifstrempty{#1}%
  {%
```

String is empty so set to −1.

```
      #2=-1\relax
    }%
    {%
```

Do we have a word break?

```
      \ifx#1\@dtl@wordbreak\relax
        #2=`\ \relax
      \else
        \ifcat\noexpand#1\relax%
```

Argument is a control sequence, so set to 0.

```
          #2=0\relax
        \else
          \expandafter\dtl@if@two@octets#1\relax\relax\dtl@end@if@two@octets
          {%
```

Argument is a UTF8 character.

```
           \dtlsetUTFviiilccharcode{#1}{#2}%
          }%
          {%
```

Argument is a character, so set to the lower case code.

```
            \dtlsetlccharcode{#1}{#2}%
          }%
```

If the result is zero, which means the character doesn't have a lower case equivalent. So set to the character code.

```
          \ifnum#2=0\relax
            #2=`#1\relax
          \fi
        \fi
      \fi
    }%
  }
```

\dtlicompare

```
\dtlicompare{⟨count⟩}{⟨string1⟩}{⟨string2⟩}
```

As \dtlcompare but ignores case.

```
  \newcommand*{\dtlicompare}[3]{%
    \dtl@subnobrsp{#2}{\@dtl@argA}%
    \dtl@subnobrsp{#3}{\@dtl@argB}%
    \ifdefempty{\@dtl@argA}%
    {%
      \ifdefempty{\@dtl@argB}%
      {%
```

Both are empty, so they are equal.

```
    #1=0\relax
  }%
  {%
```

The first string is empty, but the second isn't. Therefore the first string is less than the second string.

```
    #1=-1\relax
  }%
}%
{%
  \ifdefempty{\@dtl@argB}%
  {%
```

The second string is empty, but the first isn't. Therefore the first string is greater than the second string.

```
    #1=1\relax
  }%
  {%
```

Identify all word breaks.

```
    \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
    \let\@dtl@argA\dtl@string
    \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
    \let\@dtl@argB\dtl@string
```

Get the first object and the remaining text.

```
    \expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
    \let\dtl@firstA=\dtl@first
    \let\dtl@restA=\dtl@rest
    \expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
    \let\dtl@firstB=\dtl@first
    \let\dtl@restB=\dtl@rest
```

Is the first object of ⟨*string1*⟩ a single character or a group?

```
    \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstA}%
    {%
```

It's a single character. Is the first object of ⟨*string2*⟩ a single character or a group?

```
      \expandafter\dtl@ifsingleorUTFviii\expandafter{\dtl@firstB}%
      {%
```

Both are a single character. Get the lower case character code.

```
        \expandafter\dtl@setlccharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
        \expandafter\dtl@setlccharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
```

If control sequences should be skipped, check if either is a control sequence (the character code will be 0).

```
        \let\dtl@donextcompare\@firstofone
        \ifdtlcompareskipcs
          \ifnum\dtl@codeA=0\relax
            \ifnum\dtl@codeB=0\relax
```

Skip both.

```
          \edef\dtl@donext{%
            \noexpand\dtlicompare
            {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \else
```

Skip A.

```
          \edef\dtl@donext{%
            \noexpand\dtlicompare
            {\noexpand#1}{\expandonce\dtl@restA}{\expandonce\@dtl@argB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \fi
      \else
        \ifnum\dtl@codeB=0\relax
```

Skip B.

```
          \edef\dtl@donext{%
            \noexpand\dtlicompare
            {\noexpand#1}{\expandonce\@dtl@argA}{\expandonce\dtl@restB}}%
          \dtl@donext
          \let\dtl@donextcompare\@gobble
        \fi
      \fi
    \fi
    \dtl@donextcompare
    {%
      \ifnum\dtl@codeA=-1\relax
        \ifnum\dtl@codeB=-1\relax
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
          \edef\dtl@donext{%
            \noexpand\dtlicompare{\noexpand#1}%
            {\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
          \dtl@donext
        \else
          \edef\dtl@donext{%
            \noexpand\dtlicompare
              {\noexpand#1}%
              {\expandonce\dtl@restA}%
              {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
          \dtl@donext
        \fi
      \else
        \ifnum\dtl@codeB=-1\relax
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
          \edef\dtl@donext{%
            \noexpand\dtlicompare
            {\noexpand#1}%
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@restB}}%
         \dtl@donext
      \else
        \ifnum\dtl@codeA<\dtl@codeB
          #1=-1\relax
        \else
          \ifnum\dtl@codeA>\dtl@codeB
            #1=1\relax
          \else
             \ifdefempty{\dtl@restA}%
             {%
               \ifdefempty{\dtl@restB}%
               {%
                 #1=0\relax
               }%
               {%
                 #1=-1\relax
               }%
             }%
             {%
               \ifdefempty{\dtl@restB}%
               {%
                 #1=1\relax
               }%
               {%
```

v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
                 \edef\dtl@donext{%
                   \noexpand\dtlicompare
                     {\noexpand#1}%
                     {\expandonce\dtl@restA}%
                     {\expandonce\dtl@restB}}%
                 \dtl@donext
               }%
             }%
          \fi
        \fi
      \fi
    \fi
  }%
}%
{%
```

The first object in ⟨*string1*⟩ is a single character, but the first object in ⟨*string2*⟩ isn't a single character. v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
  \edef\dtl@donext{%
    \noexpand\dtlicompare
```

```
              {\noexpand#1}%
              {\expandonce\dtl@firstA\expandonce\dtl@restA}%
              {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
          \dtl@donext
        }%
      }%
      {%
```

Neither object is a single character. v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
        \edef\dtl@donext{%
          \noexpand\dtlicompare
            {\noexpand#1}%
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
        \dtl@donext
      }%
    }%
  }%
}
```

ordindexcompare

Word breaks come before all other letters of the alphabet.

```
\newcommand*{\dtlwordindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{\@dtl@wordbreak}%
}
```

terindexcompare

Word breaks are ignored.

```
\newcommand*{\dtlletterindexcompare}[3]{%
  \@dtldictcompare{#1}{#2}{#3}{}%
}
```

@dtldictcompare

Word or letter compare. Fourth argument should be \@dtl@wordbreak for word compare or empty for letter compare.

```
\newcommand*{\@dtldictcompare}[4]{%
  \dtl@subnobrsp{#2}{\@dtl@argA}%
  \dtl@subnobrsp{#3}{\@dtl@argB}%
  \ifdefempty{\@dtl@argA}%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=0\relax
    }%
    {%
      #1=-1\relax
    }%
  }%
  {%
    \ifdefempty{\@dtl@argB}%
    {%
      #1=1\relax
    }%
    {%
```

Alphabetizing continues until a comma indicates inverted order. This assumes that an actual comma indicates that the comma forms part of the text (e.g. in a title of a play). Inversion commas should be indicated using commands such as \datatoolpersoncomma. There are three types of inverted order: people, places and subjects (concepts or objects). We also need to treat parenthetical material in a similar way. Find if the first string has an inverted order.

```
    \expandafter\DTLsplitstring\expandafter
      {\@dtl@argA}{\datatoolpersoncomma}{\@dtl@beforepart}{\@dtl@afterpart}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \expandafter\DTLsplitstring\expandafter
        {\@dtl@argA}{\datatoolplacecomma}{\@dtl@beforepart}{\@dtl@afterpart}%
      \ifdefempty{\@dtl@replaced}%
      {%
        \expandafter\DTLsplitstring\expandafter
          {\@dtl@argA}{\datatoolsubjectcomma}{\@dtl@beforepart}{\@dtl@afterpart}%
        \ifdefempty{\@dtl@replaced}%
        {%
          \expandafter\DTLsplitstring\expandafter
            {\@dtl@argA}{\datatoolparenstart}{\@dtl@beforepart}{\@dtl@afterpart}%
          \ifdefempty{\@dtl@replaced}%
          {%
            \def\@dtl@A@comma{0}%
            \let\@dtl@A@before\@dtl@argA
            \def\@dtl@A@after{}%
          }%
          {%
            \let\@dtl@A@comma\@dtl@replaced
            \let\@dtl@A@before\@dtl@beforepart
            \let\@dtl@A@after\@dtl@afterpart
          }%
        }%
        {%
          \let\@dtl@A@comma\@dtl@replaced
          \let\@dtl@A@before\@dtl@beforepart
          \let\@dtl@A@after\@dtl@afterpart
        }%
      }%
      {%
        \let\@dtl@A@comma\@dtl@replaced
        \let\@dtl@A@before\@dtl@beforepart
        \let\@dtl@A@after\@dtl@afterpart
      }%
    }%
    {%
      \let\@dtl@A@comma\@dtl@replaced
      \let\@dtl@A@before\@dtl@beforepart
      \let\@dtl@A@after\@dtl@afterpart
    }%
```

Now find if the second string has an inverted order.

```
  \expandafter\DTLsplitstring\expandafter
    {\@dtl@argB}{\datatoolpersoncomma}{\@dtl@beforepart}{\@dtl@afterpart}%
  \ifdefempty{\@dtl@replaced}%
  {%
    \expandafter\DTLsplitstring\expandafter
      {\@dtl@argB}{\datatoolplacecomma}{\@dtl@beforepart}{\@dtl@afterpart}%
    \ifdefempty{\@dtl@replaced}%
    {%
      \expandafter\DTLsplitstring\expandafter
        {\@dtl@argB}{\datatoolsubjectcomma}{\@dtl@beforepart}{\@dtl@afterpart}%
      \ifdefempty{\@dtl@replaced}%
      {%
        \expandafter\DTLsplitstring\expandafter
          {\@dtl@argB}{\datatoolparenstart}{\@dtl@beforepart}{\@dtl@afterpart}%
        \ifdefempty{\@dtl@replaced}%
        {%
          \def\@dtl@B@comma{0}%
          \let\@dtl@B@before\@dtl@argB
          \def\@dtl@B@after{}%
        }%
        {%
          \let\@dtl@B@comma\@dtl@replaced
          \let\@dtl@B@before\@dtl@beforepart
          \let\@dtl@B@after\@dtl@afterpart
        }%
      }%
      {%
        \let\@dtl@B@comma\@dtl@replaced
        \let\@dtl@B@before\@dtl@beforepart
        \let\@dtl@B@after\@dtl@afterpart
      }%
    }%
    {%
      \let\@dtl@B@comma\@dtl@replaced
      \let\@dtl@B@before\@dtl@beforepart
      \let\@dtl@B@after\@dtl@afterpart
    }%
  }%
  {%
    \let\@dtl@B@comma\@dtl@replaced
    \let\@dtl@B@before\@dtl@beforepart
    \let\@dtl@B@after\@dtl@afterpart
  }%
```

Get the first letter and find out if it’s a letter, digit or symbol.

```
\expandafter\dtl@ifcasechargroup\@dtl@A@before\dtl@end@ifcasechargroup
 {\def\@dtl@A@chargroup{2}}%
 {\def\@dtl@A@chargroup{1}}%
 {\def\@dtl@A@chargroup{0}}%
\expandafter\dtl@ifcasechargroup\@dtl@B@before\dtl@end@ifcasechargroup
 {\def\@dtl@B@chargroup{2}}%
 {\def\@dtl@B@chargroup{1}}%
 {\def\@dtl@B@chargroup{0}}%
```

Are they in the same group?

```
\ifnum\@dtl@A@chargroup<\@dtl@B@chargroup
  #1=-1\relax
\else
  \ifnum\@dtl@A@chargroup>\@dtl@B@chargroup
    #1=1\relax
  \else
```

In the same group. Which group are they in?

```
    \ifcase\@dtl@A@chargroup
```

Symbol group v2.25: added \expandonce to prevent non-ASCII characters from being expanded.

```
      \edef\dtl@donext{%
        \noexpand\dtlcompare
          {\noexpand#1}%
          {\expandonce\@dtl@A@before}%
          {\expandonce\@dtl@B@before}}%
      \dtl@donext
```

Number.

```
    \or
      \ifnum\@dtl@A@before<\@dtl@B@before\relax
        #1=-1\relax
      \else
        \ifnum\@dtl@A@before>\@dtl@B@before\relax
          #1=1\relax
        \else
          #1=0\relax
        \fi
      \fi
    \or
```

Word or phrase.

```
      \@dtlwordindexcompare{#1}{\@dtl@A@before}{\@dtl@B@before}
        {\dtlicomparewords}{#4}%
```

If they are equal, do we have an inverted order?

```
      \ifnum#1=0\relax
```

Temporarily redefine the inversion commas to numbers to make the comparisons easier.

```
        \let\@org@dtl@person@comma\datatoolpersoncomma
        \let\@org@dtl@place@comma\datatoolplacecomma
```

```
              \let\@org@dtl@subject@comma\datatoolsubjectcomma
              \let\@org@dtl@paren@start\datatoolparenstart
```

People first, then places, then subjects, then no inversion, then parenthetical.

```
              \def\datatoolpersoncomma{3}%
              \def\datatoolplacecomma{2}%
              \def\datatoolsubjectcomma{1}%
              \def\datatoolparenstart{-1}%
```

Now compare:

```
              \ifnum\@dtl@A@comma>\@dtl@B@comma\relax
                #1=-1\relax
              \else
                \ifnum\@dtl@A@comma<\@dtl@B@comma\relax
                  #1=1\relax
                \else
```

They are the same type. First do a reverse case sensitive comparison.

```
                 \@dtlwordindexcompare{#1}{\@dtl@B@before}{\@dtl@A@before}
                   {\dtlcomparewords}{#4}%
```

Are they still equal?

```
                  \ifnum#1=0\relax
```

So sort on inversion.

```
                    \@dtlwordindexcompare{#1}{\@dtl@A@after}{\@dtl@B@after}
                      {\dtlicomparewords}{#4}%
                  \fi
                \fi
              \fi
```

Restore original definitions.

```
              \let\datatoolpersoncomma\@org@dtl@person@comma
              \let\datatoolplacecomma\@org@dtl@place@comma
              \let\datatoolsubjectcomma\@org@dtl@subject@comma
              \let\datatoolparenstart\@org@dtl@paren@start
            \fi
          \fi
        \fi
      \fi
    }%
  }%
}%
```

Need to indicate type of inversion.

toolpersoncomma

```
\newcommand*{\datatoolpersoncomma}{,\space}
```

atoolplacecomma

```
\newcommand*{\datatoolplacecomma}{,\space}
```

oolsubjectcomma

```
\newcommand*{\datatoolsubjectcomma}{,\space}
```

atoolparenstart

```
\newcommand*{\datatoolparenstart}{\space}
```

wordindexcompare

```
\@dtlwordindexcompare{⟨count⟩}{⟨cs A⟩}{⟨cs B⟩}{⟨word comparison handler⟩}
{⟨word break replacement⟩}
```

```
\newcommand*{\@dtlwordindexcompare}[5]{%
```

Word or phrase. Replace word breaks.

```
  \dtl@setwordbreaks{#2}{#5}%
  \let#2\dtl@string
```

And again for the second string.

```
  \dtl@setwordbreaks{#3}{}%
  \let#3\dtl@string
```

Now compare both strings.

```
%  \@dtl@dict@compare{#1}{#2}{#3}{#4}%
  \edef\@dtl@do@compare{%
    \noexpand#4{\noexpand#1}%
      {\expandonce#2}{\expandonce#3}%
  }%
  \@dtl@do@compare
}
```

dtl@dict@compare

```
\@dtl@dict@compare{⟨count⟩}{⟨cs A⟩}{⟨cs B⟩}{⟨word comparison handler⟩}
```

Now that all the word breaks have been identified with `\@dtl@wordbreak` compare both strings.

```
\newcommand*{\@dtl@dict@compare}[4]{%
```

Are either empty?

```
  \ifdefempty{#2}%
  {%
```

A is empty. Is B empty?

```
    \ifdefempty{#3}%
    {%
```

Both are empty.

```
      #1=0\relax
    }%
    {%
```

A is empty but B isn't

```
      #1=-1\relax
    }%
  }%
  {%
```

A isn't empty. Is B empty?

```
    \ifdefempty{#3}%
    {%
```

B is empty but A isn't.

```
      #1=1\relax
    }%
    {%
```

Neither are empty. Grab first word from A.

```
      \expandafter\dtl@grabword#2\@dtl@endgrabword\dtl@A@first\dtl@A@remain
```

Grab first word from B.

```
      \expandafter\dtl@grabword#3\@dtl@endgrabword\dtl@B@first\dtl@B@remain
```

Compare A and B.

```
      \edef\@dtl@do@compare{%
        \noexpand#4{\noexpand#1}%
          {\expandonce\dtl@A@first}{\expandonce\dtl@B@first}%
      }%
      \@dtl@do@compare
```

Are they the same?

```
      \ifnum#1=0\relax
```

They are, so compare on the next word.

```
        \@dtl@dict@compare{#1}{\dtl@A@remain}{\dtl@B@remain}{#4}%
      \fi
    }%
  }%
}
```

\dtl@grabword Grab first word from phrase.

```
\def\dtl@grabword#1\@dtl@wordbreak#2\@dtl@endgrabword#3#4{%
  \def#3{#1}%
  \def#4{#2}%
}
```

dtlicomparewords

\dtlicomparewords{⟨*count*⟩}{⟨*word A*⟩}{⟨*word B*⟩}

This does a case insensitive comparison.

```
\newcommand{\dtlicomparewords}[3]{%
  \dtlicompare{#1}{#2}{#3}%
}
```

\dtlcomparewords

```
\dtlcomparewords{⟨count⟩}{⟨word A⟩}{⟨word B⟩}
```

This does a case sensitive comparison.

```
\newcommand{\dtlcomparewords}[3]{%
  \dtlcompare{#1}{#2}{#3}%
}
```

l@setwordbreaks

Replace word breaks (space, \space, \, ~ and hyphen -) with the second argument (either \@dtl@wordbreak for letter sort or nothing for word sort). Result is stored in \dtl@string.

```
\newcommand*{\dtl@setwordbreaks}[2]{%
  \expandafter\dtl@subnobrsp\expandafter{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{#2}%
  \DTLsubstituteall{\dtl@string}{\ }{#2}%
  \DTLsubstituteall{\dtl@string}{\space}{#2}%
  \DTLsubstituteall{\dtl@string}{-}{#2}%
```

Now deal with actual spaces.

```
 \toks@{#2}%
 \edef\dtl@do@setwordbreaks{%
   \noexpand\@dtl@setwordbreaks{\the\toks@}\expandonce\dtl@string\space\noexpand\@nil}%
 \def\dtl@string{}%
 \dtl@do@setwordbreaks
}
```

l@setwordbreaks

```
\def\@dtl@setwordbreaks#1#2 #3{%
  \def\dtl@tmp{#3}%
  \ifx\@nnil\dtl@tmp
```

Reached end of loop.

```
    \let\@dtl@setwordbreaks@next\@gobbletwo
    \appto\dtl@string{#2}%
  \else
    \let\@dtl@setwordbreaks@next\@dtl@setwordbreaks
    \appto\dtl@string{#2#1}%
  \fi
  \@dtl@setwordbreaks@next{#1}#3%
}
```

breaksnohyphens

As \dtl@setwordbreaks but excludes hyphens.

```
\newcommand*{\dtl@setwordbreaksnohyphens}[2]{%
  \expandafter\dtl@subnobrsp\expandafter{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{#2}%
  \DTLsubstituteall{\dtl@string}{\ }{#2}%
  \DTLsubstituteall{\dtl@string}{\space}{#2}%
```

Now deal with actual spaces.

```
 \toks@{#2}%
 \edef\dtl@do@setwordbreaks{%
   \noexpand\@dtl@setwordbreaks{\the\toks@}\expandonce\dtl@string\space\noexpand\@nil}%
 \def\dtl@string{}%
 \dtl@do@setwordbreaks
}
```

\@dtl@wordbreak

```
\newcommand*{\@dtl@wordbreak}{ }
```

ifcasechargroup Determine if first character is a letter, a digit or a symbol.

```
\def\dtl@ifcasechargroup#1#2\dtl@end@ifcasechargroup#3#4#5{%
```

Does it start with a UTF8 character?

```
 \expandafter\dtl@if@two@octets#1#2\relax\relax\dtl@end@if@two@octets
 {%
```

Get the lower case character code.

```
   \dtl@getfirst@UTFviii#1#2\@nil\end@dtl@getfirst@UTFviii
   \expandafter\dtlsetUTFviiilccharcode\expandafter{\dtl@first}{\count@}%
   \ifnum\count@<`a\relax #5\else#3\fi
 }%
 {%
  \dtlifcasechargroup{#1}%
   {#3}%
   {%
```

Starts with a digit. Is the whole thing an integer?

```
     \DTLifint{#1#2}
     {%
       #4%
     }%
     {%
```

No, it isn't. Consider it a string.

```
       #3%
     }%
   }%
   {#5}%
 }%
}
```

macro \dtlifcasechargroup{⟨char⟩}{⟨case letter⟩}{⟨case digit⟩}{⟨case symbol⟩}

```
\newcommand*{\dtlifcasechargroup}[4]{%
  \count@=`#1\relax
  \dtlifintclosedbetween{\number\count@}{48}{57}%
  {%
```

It’s a digit.

```
    #3%
  }%
  {%
    \dtlifintclosedbetween{\number\count@}{97}{122}%
    {%
```

Lower case letter

```
      #2%
    }%
    {%
      \dtlifintclosedbetween{\number\count@}{65}{90}%
      {%
```

Upper case letter

```
        #2%
      }%
      {%
```

Other

```
        #4%
      }%
    }%
  }%
}
```

\dtlparsewords

\dtlparsewords{⟨*phrase*⟩}{⟨*handler cs*⟩}

Iterates through the given phrase. Hyphens are considered word boundaries.

```
\newcommand*{\dtlparsewords}[2]{%
  \dtl@subnobrsp{#1}{\dtl@string}%
  \DTLsubstituteall{\dtl@string}{~}{ }%
  \DTLsubstituteall{\dtl@string}{\ }{ }%
  \DTLsubstituteall{\dtl@string}{\space}{ }%
  \DTLsubstituteall{\dtl@string}{-}{ }%
  \let\dtl@parsewordshandler#2\relax
  \edef\dtl@donext{%
     \noexpand\@dtl@parse@words\expandonce\dtl@string\space\noexpand\@nil}%
  \dtl@donext
}
```

dtl@parse@words

```
\def\@dtl@parse@words#1 #2{%
  \def\dtl@tmp{#2}%
  \ifx\@nnil\dtl@tmp
   \let\parse@wordsnext=\@gobble
  \else
```

```
    \let\parse@wordsnext=\@dtl@parse@words
  \fi
  \dlt@parsewordshandler{#1}%
  \parse@wordsnext#2%
}
```

\DTLifstringlt

`\DTLifstringlt{⟨string1⟩}{⟨string2⟩}{⟨true part⟩}{⟨false part⟩}`

String comparison (Starred version ignores case)

```
\newcommand*{\DTLifstringlt}{\@ifstar\@sDTLifstringlt\@DTLifstringlt}
```

Unstarred version

```
\newcommand*{\@DTLifstringlt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount<0\relax
    #3%
  \else
    #4%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifstringlt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount<0\relax
    #3%
  \else
    #4%
  \fi
}
```

\DTLiflt

`\DTLiflt{⟨arg1⟩}{⟨arg2⟩}{⟨true part⟩}{⟨false part⟩}`

Does `\DTLifnumlt` if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do `\DTLifstringlt` (unstarred version) or `\DTLifstringlt*` (starred version).

```
\newcommand*{\DTLiflt}{\@ifstar\@sDTLiflt\@DTLiflt}
```

Unstarred version

```
\newcommand*{\@DTLiflt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
```

```
   \DTLifnumlt{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringlt{#1}{#2}{#3}{#4}%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLiflt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumlt{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringlt{#1}{#2}{#3}{#4}%
  \fi
}
```

\DTLifnumgt

\DTLifnumgt{⟨*num1*⟩}{⟨*num2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if {⟨*num1*⟩} > {⟨*num2*⟩}. Both numbers need to have the decimal separator changed to a dot to ensure that it works with `\dtlifnumgt`

```
\newcommand*{\DTLifnumgt}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumgt{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
```

\DTLifstringgt

\DTLifstringgt{⟨*string1*⟩}{⟨*string2*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

String comparison (starred version ignores case)

```
\newcommand*{\DTLifstringgt}{\@ifstar\@sDTLifstringgt\@DTLifstringgt}
```

Unstarred version

```
\newcommand*{\@DTLifstringgt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount>0\relax
    #3%
  \else
    #4%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifstringgt}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount>0\relax
    #3%
  \else
    #4%
  \fi
}
```

\DTLifgt

```
\DTLifgt{⟨arg1⟩}{⟨arg2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does `\DTLifnumgt` if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do `\DTLifstringgt` or `\DTLifstringgt*`.

```
\newcommand*{\DTLifgt}{\@ifstar\@sDTLifgt\@DTLifgt}
```

Unstarred version

```
\newcommand*{\@DTLifgt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumgt{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringgt{#1}{#2}{#3}{#4}%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifgt}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumgt{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringgt{#1}{#2}{#3}{#4}%
  \fi
}
```

\DTLifnumeq

```
\DTLifnumeq{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if {⟨*num1*⟩} = {⟨*num2*⟩}. Both numbers need to have the decimal separator changed to a dot to ensure that it works with `\dtlifnumeq`

```
\newcommand*{\DTLifnumeq}[4]{%
  \DTLconverttodecimal{#1}{\@dtl@numi}%
  \DTLconverttodecimal{#2}{\@dtl@numii}%
  \dtlifnumeq{\@dtl@numi}{\@dtl@numii}%
  {%
    #3%
  }%
  {%
    #4%
  }%
}
```

\DTLifstringeq

```
\DTLifstringeq{⟨string1⟩}{⟨string2⟩}{⟨true part⟩}{⟨false part⟩}
```

String comparison (starred version ignores case)

```
\newcommand*{\DTLifstringeq}{\@ifstar\@sDTLifstringeq\@DTLifstringeq}
```

Unstarred version

```
\newcommand*{\@DTLifstringeq}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount=0\relax
    #3%
  \else
    #4%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifstringeq}[4]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \ifnum\@dtl@tmpcount=0\relax
    #3%
  \else
    #4%
  \fi
}
```

\DTLifeq

```
\DTLifeq{⟨arg1⟩}{⟨arg2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does \DTLifnumeq if both ⟨*arg1*⟩ and ⟨*arg2*⟩ are numerical, otherwise do \DTLifstringeq or \DTLifstringeq*.

```
\newcommand*{\DTLifeq}{\@ifstar\@sDTLifeq\@DTLifeq}
```

Unstarred version

```
\newcommand*{\@DTLifeq}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumeq{#1}{#2}{#3}{#4}%
  \else
   \@DTLifstringeq{#1}{#2}{#3}{#4}%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifeq}[4]{%
  \dtl@testbothnumerical{#1}{#2}%
  \if@dtl@condition
   \DTLifnumeq{#1}{#2}{#3}{#4}%
  \else
   \@sDTLifstringeq{#1}{#2}{#3}{#4}%
  \fi
}
```

\DTLifSubString

\DTLifSubString{⟨*string*⟩}{⟨*sub string*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

If ⟨*sub string*⟩ is contained in ⟨*string*⟩ does ⟨*true part*⟩, otherwise does ⟨*false part*⟩.

```
\newcommand*{\DTLifSubString}[4]{%
  \protected@edef\@dtl@dotestifsubstring{\noexpand\dtl@testifsubstring
  {#1}{#2}}%
  \@dtl@dotestifsubstring
  \if@dtl@condition
   #3%
  \else
   #4%
  \fi
}
```

testifsubstring

```
\newcommand*{\dtl@testifsubstring}[2]{%
  \dtl@subnobrsp{#1}{\@dtl@argA}%
  \dtl@subnobrsp{#2}{\@dtl@argB}%
```

Identify all word breaks.

```
  \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
  \let\@dtl@argA\dtl@string
  \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
  \let\@dtl@argB\dtl@string
  \edef\dtl@donext{%
    \noexpand\@dtl@testifsubstring{\expandonce\@dtl@argA}{\expandonce\@dtl@argB}}%
  \dtl@donext
}
\newcommand*{\@dtl@testifsubstring}[2]{%

  \def\@dtl@subs@argA{#1}%
  \def\@dtl@subs@argB{#2}%
  \ifdefempty{\@dtl@subs@argB}%
  {%
    \@dtl@conditiontrue
  }%
  {%
    \ifdefempty{\@dtl@subs@argA}%
    {%
      \@dtl@conditionfalse
    }%
    {%
      \@dtl@teststartswith{#1}{#2}%
      \if@dtl@condition
      \else
        \dtl@getfirst#1\end@dtl@getfirst
        \expandafter\dtl@ifsingle\expandafter{\dtl@first}%
        {%
          \expandafter\@dtl@testifsubstring\expandafter{\dtl@rest}{#2}%
        }%
        {%
          \protected@edef\@dtl@donext{\noexpand\@dtl@testifsubstring
             {\expandonce\dtl@first\expandonce\dtl@rest}{\expandonce\@dtl@subs@argB}}%
          \@dtl@donext
        }%
      \fi
    }%
  }%
}
```

\DTLifStartsWith `\DTLifStartsWith{⟨string⟩}{⟨substring⟩}{⟨true part⟩}{⟨false part⟩}`

If ⟨*string*⟩ starts with ⟨*substring*⟩, this does ⟨*true part*⟩, otherwise it does ⟨*false part*⟩.

```
\newcommand*{\DTLifStartsWith}[4]{%
  \@dtl@conditionfalse
  \protected@edef\@dtl@tmp{\noexpand\dtl@teststartswith{#1}{#2}}%
  \@dtl@tmp
  \if@dtl@condition
   #3%
  \else
```

```
      #4%
    \fi
  }
```

l@teststartswith

\dtl@teststartswith{⟨*string*⟩}{⟨*prefix*⟩}

Tests if ⟨*string*⟩ starts with ⟨*prefix*⟩. This sets \if@dtl@condition. First substitute all word breaks with \dtl@setwordbreaksnohyphen

```
  \newcommand*{\dtl@teststartswith}[2]{%
    \dtl@subnobrsp{#1}{\@dtl@argA}%
    \dtl@subnobrsp{#2}{\@dtl@argB}%
```

Identify all word breaks.

```
    \dtl@setwordbreaksnohyphens{\@dtl@argA}{\@dtl@wordbreak}%
    \let\@dtl@argA\dtl@string
    \dtl@setwordbreaksnohyphens{\@dtl@argB}{\@dtl@wordbreak}%
    \let\@dtl@argB\dtl@string
    \edef\dtl@donext{%
      \noexpand\@dtl@teststartswith{\expandonce\@dtl@argA}{\expandonce\@dtl@argB}}%
    \dtl@donext
  }

  \newcommand*{\@dtl@teststartswith}[2]{%
    \def\@dtl@argA{#1}%
    \def\@dtl@argB{#2}%
    \ifdefempty{\@dtl@argA}%
    {%
      \ifdefempty{\@dtl@argB}%
      {%
        \@dtl@conditiontrue
      }%
      {%
        \@dtl@conditionfalse
      }%
    }%
    {%
      \ifdefempty{\@dtl@argB}%
      {%
        \@dtl@conditiontrue
      }%
      {%
        \expandafter\dtl@getfirst\@dtl@argA\end@dtl@getfirst
```

Get the first object and the remaining text.

```
        \let\dtl@firstA=\dtl@first
        \let\dtl@restA=\dtl@rest
        \expandafter\dtl@getfirst\@dtl@argB\end@dtl@getfirst
```

```
\let\dtl@firstB=\dtl@first
\let\dtl@restB=\dtl@rest
```

Is the first object of ⟨*string1*⟩ a single character or a group?

```
\expandafter\dtl@ifsingle\expandafter{\dtl@firstA}%
{%
```

It's a single character. Is the first object of ⟨*string2*⟩ a single character or a group?

```
  \expandafter\dtl@ifsingle\expandafter{\dtl@firstB}%
  {%
```

Both are a single character. Get the lower case character code.

```
    \expandafter\dtl@setcharcode\expandafter{\dtl@firstA}{\dtl@codeA}%
    \expandafter\dtl@setcharcode\expandafter{\dtl@firstB}{\dtl@codeB}%
    \ifnum\dtl@codeA=-1\relax
      \ifnum\dtl@codeB=-1\relax
         \protected@edef\dtl@donext{%
           \noexpand\@dtl@teststartswith{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
         \dtl@donext
      \else
         \protected@edef\dtl@donext{%
           \noexpand\@dtl@teststartswith
             {\expandonce\dtl@restA}{\expandonce\dtl@firstB\expandonce\dtl@restB}}%
         \dtl@donext
      \fi
    \else
      \ifnum\dtl@codeB=-1\relax
         \protected@edef\dtl@donext{%
           \noexpand\@dtl@teststartswith
             {\expandonce\dtl@firstA\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
         \dtl@donext
      \else
        \ifnum\dtl@codeA=\dtl@codeB
          \protected@edef\dtl@donext{%
            \noexpand\@dtl@teststartswith{\expandonce\dtl@restA}{\expandonce\dtl@restB}}%
          \dtl@donext
        \else
          \@dtl@conditionfalse
        \fi
      \fi
    \fi
  }%
  {%
```

The first object in ⟨*string1*⟩ is a single character, but the first object in ⟨*string2*⟩ isn't a single character.

```
    \protected@edef\dtl@donext{%
      \noexpand\@dtl@teststartswith
        {\expandonce\dtl@firstA\expandonce\dtl@restA}%
        {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
    \dtl@donext
```

```
        }%
      }%
      {%
```

Neither object is a single character.

```
        \protected@edef\dtl@donext{%
          \noexpand\@dtl@teststartswith
            {\expandonce\dtl@firstA\expandonce\dtl@restA}%
            {\expandonce\dtl@firstB\expandonce\dtl@restB}}%
      }%
    }%
  }%
}
```

numclosedbetween

\DTLifnumclosedbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩.

```
\newcommand*{\DTLifnumclosedbetween}[5]{%
 \DTLconverttodecimal{#1}{\@dtl@numi}%
 \DTLconverttodecimal{#2}{\@dtl@numii}%
 \DTLconverttodecimal{#3}{\@dtl@numiii}%
 \DTLifFPclosedbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
```

ingclosedbetween

\DTLifstringclosedbetween{⟨*string*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

String comparison (starred version ignores case)

```
\newcommand*{\DTLifstringclosedbetween}{%
  \@ifstar\@sDTLifstringclosedbetween\@DTLifstringclosedbetween
}
```

Unstarred version

```
\newcommand*{\@DTLifstringclosedbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount<0\relax
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount>0\relax
      \def\@dtl@dovalue{#5}%
    \else
      \def\@dtl@dovalue{#4}%
    \fi
  \fi
  \@dtl@dovalue
}
```

Starred version

```
\newcommand*{\@sDTLifstringclosedbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount<0\relax
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount>0\relax
      \def\@dtl@dovalue{#5}%
    \else
      \def\@dtl@dovalue{#4}%
    \fi
  \fi
  \@dtl@dovalue
}
```

Lifclosedbetween

\DTLifclosedbetween{⟨*arg*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Does \DTLifnumclosedbetween if {⟨*arg*⟩}, ⟨*min*⟩ and ⟨*max*⟩ are numerical, otherwise do \DTLifstringclosedbetween or \DTLifstringclosedbetween*.

```
\newcommand*{\DTLifclosedbetween}{%
  \@ifstar\@sDTLifclosedbetween\@DTLifclosedbetween
}
```

Unstarred version

```
\newcommand*{\@DTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@DTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@DTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifclosedbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@sDTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@sDTLifstringclosedbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
```

ifnumopenbetween

\DTLifnumopenbetween{⟨*num*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩.

```
\newcommand*{\DTLifnumopenbetween}[5]{%
 \DTLconverttodecimal{#1}{\@dtl@numi}%
 \DTLconverttodecimal{#2}{\@dtl@numii}%
 \DTLconverttodecimal{#3}{\@dtl@numiii}%
 \DTLifFPopenbetween{\@dtl@numi}{\@dtl@numii}{\@dtl@numiii}{#4}{#5}%
}
```

tringopenbetween

\DTLifstringopenbetween{⟨*string*⟩}{⟨*min*⟩}{⟨*max*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

String comparison (starred version ignores case)

```
\newcommand*{\DTLifstringopenbetween}{%
  \@ifstar\@sDTLifstringopenbetween\@DTLifstringopenbetween
}
```

Unstarred version:

```
\newcommand*{\@DTLifstringopenbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount>0\relax
  \else
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlcompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount<0\relax
      \def\@dtl@dovalue{#4}%
    \else
      \def\@dtl@dovalue{#5}%
    \fi
  \fi
  \@dtl@dovalue
}
```

Starred version

```
\newcommand*{\@sDTLifstringopenbetween}[5]{%
  \protected@edef\@dtl@tmpcmp{%
    \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#2}}%
  \@dtl@tmpcmp
  \let\@dtl@dovalue\relax
  \ifnum\@dtl@tmpcount>0\relax
  \else
    \def\@dtl@dovalue{#5}%
  \fi
  \ifx\@dtl@dovalue\relax
    \protected@edef\@dtl@tmpcmp{%
      \noexpand\dtlicompare{\noexpand\@dtl@tmpcount}{#1}{#3}}%
    \@dtl@tmpcmp
    \ifnum\@dtl@tmpcount<0\relax
      \def\@dtl@dovalue{#4}%
    \else
      \def\@dtl@dovalue{#5}%
    \fi
  \fi
  \@dtl@dovalue
}
```

DTLifopenbetween

```
\DTLifopenbetween{⟨arg⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Does `\DTLifnumopenbetween` if {⟨*arg*⟩}, ⟨*min*⟩ and ⟨*max*⟩ are numerical, otherwise do `\DTLifstringopenbetween` or `\DTLifstringopenbetween*`.

```
\newcommand*{\DTLifopenbetween}{%
  \@ifstar\@sDTLifopenbetween\@DTLifopenbetween
}
```

Unstarred version

```
\newcommand*{\@DTLifopenbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumopenbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@DTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@DTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
```

Starred version

```
\newcommand*{\@sDTLifopenbetween}[5]{%
  \dtl@testbothnumerical{#2}{#3}%
  \if@dtl@condition
    \dtl@ifsingle{#1}{%
      \edef\@dtl@tmp{#1}}{%
      \def\@dtl@tmp{#1}}%
    \expandafter\@dtl@checknumerical\expandafter{\@dtl@tmp}%
    \ifnum\@dtl@datatype>0\relax
      \DTLifnumopenbetween{#1}{#2}{#3}{#4}{#5}%
    \else
      \@sDTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
    \fi
  \else
    \@sDTLifstringopenbetween{#1}{#2}{#3}{#4}{#5}%
  \fi
}
```

LifFPopenbetween

`\DTLifFPopenbetween{`⟨*num*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩ where all arguments are in standard fixed point notation. (Command name maintained for backward compatibility.)

```
\let\DTLifFPopenbetween\dtlifnumopenbetween
```

fFPclosedbetween

```
\DTLifFPclosedbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩. (Command name maintained for backward compatibility.)

```
\let\DTLifFPclosedbetween\dtlifnumclosedbetween
```

1.6.2 ifthen Conditionals

The following commands provide conditionals `\DTLis...` which can be used in `\ifthenelse`.

`\dtl@testlt` Command to test if first argument is less than second argument. If either argument is a string, a case sensitive string comparison is used instead. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testlt}[2]{%
  \DTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLislt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLislt}[2]{%
  \TE@throw\noexpand\dtl@testlt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testiclt` Command to test if first argument is less than second argument. If either argument is a string, a case insensitive string comparison is used instead. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testiclt}[2]{%
  \@sDTLiflt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisilt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisilt}[2]{%
  \TE@throw\noexpand\dtl@testiclt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testgt` Command to test if first argument is greater than second argument. This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testgt}[2]{%
  \DTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

`\DTLisgt` Provide conditional command for use in `\ifthenelse`

```
\newcommand*{\DTLisgt}[2]{%
  \TE@throw\noexpand\dtl@testgt{#1}{#2}\noexpand\if@dtl@condition
}
```

`\dtl@testicgt` Command to test if first argument is greater than second argument (ignores case). This sets `\if@dtl@condition`.

```
\newcommand*{\dtl@testicgt}[2]{%
  \@sDTLifgt{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisigt Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisigt}[2]{%
  \TE@throw\noexpand\dtl@testicgt{#1}{#2}\noexpand\if@dtl@condition
}
```

\dtl@testeq Command to test if first argument is equal to the second argument. This sets \if@dtl@condition.

```
\newcommand*{\dtl@testeq}[2]{%
  \DTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLiseq Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLiseq}[2]{%
  \TE@throw\noexpand\dtl@testeq{#1}{#2}\noexpand\if@dtl@condition
}
```

\dtl@testiceq Command to test if first number is equal to the second number (ignores case). This sets \if@dtl@condition.

```
\newcommand*{\dtl@testiceq}[2]{%
  \@sDTLifeq{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisieq Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisieq}[2]{%
  \TE@throw\noexpand\dtl@testiceq{#1}{#2}\noexpand\if@dtl@condition
}
```

\DTLisSubString Tests if second argument is contained in first argument.

```
\newcommand*{\DTLisSubString}[2]{%
  \TE@throw\noexpand\dtl@testifsubstring{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

\DTLisPrefix Tests if first argument starts with second argument.

```
\newcommand*{\DTLisPrefix}[2]{%
  \TE@throw\noexpand\dtl@teststartswith{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

\DTLisinlist Tests if first argument starts with second argument.

```
\newcommand*{\DTLisinlist}[2]{%
  \TE@throw\noexpand\dtl@testinlist{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

\dtl@testinlist

```
\newcommand*{\dtl@testinlist}[2]{%
  \DTLifinlist{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

umclosedbetween Command to test if first number lies between second and third numbers. (End points included, all arguments are fixed point numbers in standard format.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testnumclosedbetween}[3]{%
  \DTLifnumclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

Provide conditional command for use in \ifthenelse

umclosedbetween

```
\newcommand*{\DTLisnumclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

tnumopenbetween Command to test if first number lies between second and third numbers. (End points excluded, all arguments are fixed point numbers in standard format.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testnumopenbetween}[3]{%
  \DTLifnumopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

snumopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisnumopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testnumopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

stclosedbetween Command to test if first value lies between second and third values. (End points included, case sensitive.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testclosedbetween}[3]{%
  \DTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

isclosedbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

ticlosedbetween Command to test if first value lies between second and third values. (End points included, case ignored.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testiclosedbetween}[3]{%
  \@sDTLifclosedbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

siclosedbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisiclosedbetween}[3]{%
  \TE@throw\noexpand\dtl@testiclosedbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

testopenbetween Command to test if first value lies between second and third values. (End points excluded, case sensitive.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testopenbetween}[3]{%
  \DTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

TLisopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

estiopenbetween Command to test if first value lies between second and third values. (End points excluded, case ignored.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testiopenbetween}[3]{%
  \@sDTLifopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

Lisiopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisiopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testiopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

FPclosedbetween Keep old command name for backwards compatibility:

```
\let\DTLisFPclosedbetween\DTLisnumclosedbetween
```

testopenbetween Command to test if first number lies between second and third numbers. (End points excluded, all arguments are fixed point numbers in standard format.) This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPopenbetween}[3]{%
  \DTLifFPopenbetween{#1}{#2}{#3}%
    {\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

isFPopenbetween Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPopenbetween}[3]{%
  \TE@throw\noexpand\dtl@testFPopenbetween{#1}{#2}{#3}%
  \noexpand\if@dtl@condition
}
```

\dtl@testFPislt Command to test if first number is less than second number where both numbers are in standard format. This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPislt}[2]{%
  \dtlifnumlt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
    \@dtl@conditionfalse
  }%
}
```

\DTLisFPlt Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPlt}[2]{%
  \TE@throw\noexpand\dtl@testFPislt{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

\dtl@testFPisgt Command to test if first number is greater than second number where both numbers are in standard format. This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPisgt}[2]{%
  \dtlifnumgt{#1}{#2}%
  {%
    \@dtl@conditiontrue
  }%
  {%
   \@dtl@conditionfalse
  }%
}
```

\DTLisFPgt Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPgt}[2]{%
  \TE@throw\noexpand\dtl@testFPisgt{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

\dtl@testFPiseq Command to test if two numbers are equal, where both numbers are in standard decimal format

```
\newcommand*{\dtl@testFPiseq}[2]{%
 \dtlifnumeq{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
  \@dtl@conditionfalse
 }%
}
```

\DTLisFPeq Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPeq}[2]{%
  \TE@throw\noexpand\dtl@testFPiseq{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

tl@testFPislteq Command to test if first number is less than or equal to second number where both numbers are in standard format. This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPislteq}[2]{%
 \dtlifnumlt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
```

\DTLisFPlteq Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPlteq}[2]{%
 \TE@throw\noexpand\dtl@testFPislteq{#1}{#2}%
 \noexpand\if@dtl@condition
}
```

tl@testFPisgteq Command to test if first number is greater than or equal to second number where both numbers are in standard format. This sets \if@dtl@condition.

```
\newcommand*{\dtl@testFPisgteq}[2]{%
 \dtlifnumgt{#1}{#2}%
 {%
   \@dtl@conditiontrue
 }%
 {%
   \@dtl@conditionfalse
 }%
 \if@dtl@condition
 \else
  \dtl@testFPiseq{#1}{#2}%
 \fi
}
```

\DTLisFPgteq Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisFPgteq}[2]{%
  \TE@throw\noexpand\dtl@testFPisgteq{#1}{#2}%
  \noexpand\if@dtl@condition}
```

\dtl@teststring Command to test if argument is a string. This sets \if@dtl@condition

```
\newcommand*{\dtl@teststring}[1]{%
  \DTLifstring{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLisstring Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisstring}[1]{%
  \TE@throw\noexpand\dtl@teststring{#1}\noexpand\if@dtl@condition}
```

l@testnumerical Command to test if argument is a numerical. This sets \if@dtl@condition

```
\newcommand*{\dtl@testnumerical}[1]{%
  \DTLifnumerical{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}%
}
```

\DTLisnumerical Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisnumerical}[1]{%
  \TE@throw\noexpand\dtl@testnumerical{#1}\noexpand\if@dtl@condition}
```

\dtl@testint Command to test if argument is an integer. This sets \if@dtl@condition

```
\newcommand*{\dtl@testint}[1]{%
  \DTLifint{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLisint Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisint}[1]{%
  \TE@throw\noexpand\dtl@testint{#1}\noexpand\if@dtl@condition}
```

\dtl@testreal Command to test if argument is a real. This sets \if@dtl@condition

```
\newcommand*{\dtl@testreal}[1]{%
  \DTLifreal{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLisreal Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLisreal}[1]{%
  \TE@throw\noexpand\dtl@testreal{#1}\noexpand\if@dtl@condition}
```

tl@testcurrency Command to test if argument is a currency. This sets \if@dtl@condition

```
\newcommand*{\dtl@testcurrency}[1]{%
  \DTLifcurrency{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLiscurrency Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLiscurrency}[1]{%
  \TE@throw\noexpand\dtl@testcurrency{#1}\noexpand\if@dtl@condition}
```

estcurrencyunit Command to test if argument is a currency with given unit. This sets \if@dtl@condition

```
\newcommand*{\dtl@testcurrencyunit}[2]{%
  \DTLifcurrencyunit{#1}{#2}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

Liscurrencyunit Provide conditional command for use in \ifthenelse

```
\newcommand*{\DTLiscurrencyunit}[2]{%
  \TE@throw\noexpand\dtl@testcurrencyunit{#1}{#2}%
  \noexpand\if@dtl@condition
}
```

1.7 Loops

\dtlbreak Break out of loop at the end of current iteration.

```
  \newcommand*{\dtlbreak}{%
    \PackageError{datatool}{Can't break out of anything}{}%
  }
```

\dtlforint

```
\dtlforint⟨ct⟩=⟨start⟩\to⟨end⟩\step ⟨inc⟩\do{⟨body⟩}
```

⟨*ct*⟩ is a count register, ⟨*start*⟩, ⟨*end*⟩ and ⟨*inc*⟩ are integers. Group if nested or use \dtlgforint. An infinite loop may result if ⟨*inc*⟩= 0 and ⟨*start*⟩ ≤ ⟨*end*⟩ and \dtlbreak isn't used.

```
  \long\def\dtlforint#1=#2\to#3\step#4\do#5{%
```

Make a copy of old version of break function

```
    \let\@dtl@orgbreak\dtlbreak
    \def\@dtl@endloophook{}%
```

Setup break function for the loop (sets ⟨*ct*⟩ to ⟨*end*⟩ at the end of the current iteration).

```
    \def\dtlbreak{\def\@dtl@endloophook{#1=#3}}%
```

Initialise ⟨*ct*⟩

```
    #1=#2\relax
```

Check if the steps are positive or negative.

```
    \ifnum#4<0\relax
```

Counting down

```
      \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
      {%
        #5%
        \@dtl@endloophook
        \advance#1 by #4\relax
      }%
    \else
```

Counting up

```
      \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
      {%
        #5%
        \@dtl@endloophook
        \advance#1 by #4\relax
      }%
    \fi
```

Restore break function.

```
    \let\dtlbreak\@dtl@orgbreak
  }
```

l@foreach@level Count register to keep track of global nested loops.

```
\newcount\@dtl@foreach@level
```

\dtlgforint

```
\dtlgforint⟨ct⟩=⟨start⟩\to⟨end⟩\step ⟨inc⟩\do{⟨body⟩}
```

⟨*ct*⟩ is a count register, ⟨*start*⟩, ⟨*end*⟩ and ⟨*inc*⟩ are integers. An infinite loop may result if ⟨*inc*⟩=0 and ⟨*start*⟩ ≤ ⟨*end*⟩ and `\dtlbreak` isn't used.

```
\long\def\dtlgforint#1=#2\to#3\step#4\do#5{%
```

Initialise

```
  \global#1=#2\relax
```

Increment level counter to allow for nested loops

```
  \global\advance\@dtl@foreach@level by 1\relax
```

Set up end loop hook

```
  \expandafter\global\expandafter
    \let\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
    \relax
```

Set up the break function: Copy current definition

```
  \expandafter\global\expandafter
    \let\csname @dtl@break@\the\@dtl@foreach@level\endcsname
    \dtlbreak
```

Set up definition for this level (sets <ct> to <end> at the end of the current iteration).

```
  \gdef\dtlbreak{\expandafter
     \gdef\csname @dtl@endhook@\the\@dtl@foreach@level\endcsname{%
       #1=#3}}%
```

check the direction

```
  \ifnum#4<0\relax
```

Counting down

```
    \whiledo{\(#1>#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \else
```

Counting up (or 0 increments)

```
    \whiledo{\(#1<#3\)\TE@or\(#1=#3\)}%
    {%
      #5%
      \csname @dtl@endhook@\the\@dtl@foreach@level\endcsname
      \global\advance#1 by #4\relax
    }%
  \fi
```

Restore break function

```
  \expandafter\global\expandafter\let\expandafter\dtlbreak
    \csname @dtl@break@\the\@dtl@foreach@level\endcsname
```

Decrement level counter

```
  \global\advance\@dtl@foreach@level by -1\relax
}
```

dtlenvgforint Environment form (contents are gathered, so verbatim can't be used):

```
\newenvironment{dtlenvgforint}[1]%
{%
  \def\@dtlenvgforint@arg{#1}%
  \long@collect@body\@do@dtlenvgforint
}%
{}
\newcommand{\@do@dtlenvgforint}[1]{%
  \expandafter\dtlgforint\@dtlenvgforint@arg\do{#1}%
}
```

2 datatool-fp.sty

Definitions of fixed-point commands that use the fp package.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-fp}[2019/09/27 v2.32 (NLCT)]
```

Required packages:

```
\RequirePackage{xkeyval}
\RequirePackage{fp}
\RequirePackage{datatool-base}
```

verbose Switch fp messages on or off

```
\define@choicekey{datatool-fp}{verbose}[\val\nr]{true,false}[true]{%
  \ifcase\nr\relax
    \FPmessagestrue
  \or
    \FPmessagesfalse
  \fi
}
\let\ifFPmessages\ifdtlverbose
```

Process package options:

```
\ProcessOptionsX
```

Define commands that are needed before loading datatool-base:

```
\providecommand*{\@dtl@mathprocessor}{fp}
```

\dtlifnumeq

```
\dtlifnumeq{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newcommand*{\dtlifnumeq}[4]{%
  \FPifeq{#1}{#2}%
    #3%
  \else
    #4%
  \fi
}
```

If verbose option set, switch on verbose for datatool-base as well:

```
\let\ifdtlverbose\ifFPmessages
```

2.1 Comparison Commands

\dtlifnumlt

```
\dtlifnumlt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newcommand*{\dtlifnumlt}[4]{%
  \FPiflt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
```

\dtlifnumgt

```
\dtlifnumgt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newcommand*{\dtlifnumgt}[4]{%
  \FPifgt{#1}{#2}%
   #3%
  \else
   #4%
  \fi
}
```

ifnumopenbetween

```
\dtlifnumopenbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩ where all arguments are in standard fixed point notation.

```
\newcommand*{\dtlifnumopenbetween}[5]{%
  \let\@dtl@dovalue\relax
  \dtlifnumgt{#1}{#2}%
  {}%
  {%
    \def\@dtl@dovalue{#5}%
  }%
  \dtlifnumlt{#1}{#3}%
  {%
    \ifx\@dtl@dovalue\relax
```

```
        \def\@dtl@dovalue{#4}%
      \fi
    }%
    {%
      \def\@dtl@dovalue{#5}%
    }%
    \@dtl@dovalue
  }
```

numclosedbetween

```
\dtlifnumclosedbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩ where all arguments are in standard fixed point notation.

```
  \newcommand*{\dtlifnumclosedbetween}[5]{%
    \let\@dtl@dovalue\relax
    \dtlifnumgt{#1}{#2}%
    {}%
    {%
      \dtlifnumeq{#1}{#2}%
      {%
        \def\@dtl@dovalue{#4}%
      }%
      {%
        \def\@dtl@dovalue{#5}%
      }%
    }%
    \dtlifnumlt{#1}{#3}%
    {%
      \ifx\@dtl@dovalue\relax
        \def\@dtl@dovalue{#4}%
      \fi
    }%
    {%
      \dtlifnumeq{#1}{#3}%
      {%
        \def\@dtl@dovalue{#4}%
      }%
      {%
        \def\@dtl@dovalue{#5}%
      }%
    }%
    \@dtl@dovalue
  }
```

2.2 Functions

\dtladd Adds two numbers using fp.

```
\newcommand*{\dtladd}[3]{%
  \FPadd{#1}{#2}{#3}%
}
```

\dtlsub Subtracts two numbers using fp.

```
\newcommand*{\dtlsub}[3]{%
  \FPsub{#1}{#2}{#3}%
}
```

\dtlmul Multiplies two numbers using fp.

```
\newcommand*{\dtlmul}[3]{%
  \FPmul{#1}{#2}{#3}%
}
```

\dtldiv Divides two numbers using fp.

```
\newcommand*{\dtldiv}[3]{%
  \FPdiv{#1}{#2}{#3}%
}
```

\dtlroot Square root using fp.

```
\newcommand*{\dtlroot}[2]{%
  \FProot{#1}{#2}%
}
```

\dtlround Rounds using fp.

```
\newcommand*{\dtlround}[3]{%
  \FPround{#1}{#2}{#3}%
}
```

\dtltrunc Truncates using fp. (Third argument is the number of digits.)

```
\newcommand*{\dtltrunc}[3]{%
  \FPtrunc{#1}{#2}{#3}%
}
```

\dtlclip

```
\newcommand*{\dtlclip}[2]{%
  \FPclip{#1}{#2}%
}
```

\dtlmin Minimum of two numbers using fp.

```
\newcommand*{\dtlmin}[3]{%
  \FPmin{#1}{#2}{#3}%
}
```

\dtlmax Maximum of two numbers using fp.

```
\newcommand*{\dtlmax}[3]{%
  \FPmax{#1}{#2}{#3}%
}
```

\dtlabs Absolute value using fp.

```
\newcommand*{\dtlabs}[2]{%
  \FPabs{#1}{#2}%
}
```

\dtlneg Negative of a value using fp.

```
\newcommand*{\dtlneg}[2]{%
  \FPneg{#1}{#2}%
}
```

3 datatool-pgfmath.sty

Definitions of fixed-point commands that use the pgfmath package.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool-pgfmath}[2019/09/27 v2.32 (NLCT)]
```

Required packages:

```
\RequirePackage{xkeyval}
\RequirePackage{pgfrcs,pgfkeys,pgfmath}
```

Process package options:

```
\ProcessOptionsX
```

Define commands that are needed before loading datatool-base:

```
\providecommand*{\@dtl@mathprocessor}{pgfmath}
```

\dtlifnumeq

```
\dtlifnumeq{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩=⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator. The `\number0` part allows an empty argument to be treated as zero. (`\number` required to prevent a zero prefix indicating an octal number.)

```
\newcommand*{\dtlifnumeq}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1==\number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
```

Load base package:

```
\RequirePackage{datatool-base}
```

3.1 Comparison Commands

\dtlifnumlt

```
\dtlifnumlt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ <⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newcommand*{\dtlifnumlt}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1 < \number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
```

\dtlifnumgt

```
\dtlifnumgt{⟨num1⟩}{⟨num2⟩}{⟨true part⟩}{⟨false part⟩}
```

Does ⟨*true part*⟩ if ⟨*num1*⟩ >⟨*num2*⟩, otherwise does ⟨*false part*⟩. The numbers must use a full stop as the decimal character and no number group separator.

```
\newcommand*{\dtlifnumgt}[4]{%
  \def\@dtl@truepart{#3}%
  \def\@dtl@falsepart{#4}%
  \pgfmathifthenelse{\number0#1 > \number0#2}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
```

ifnumopenbetween

```
\dtlifnumopenbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if ⟨*min*⟩ < ⟨*num*⟩ < ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\newcommand*{\dtlifnumopenbetween}[5]{%
  \def\@dtl@truepart{#4}%
  \def\@dtl@falsepart{#5}%
  \pgfmathifthenelse
    {(\number0#2 < \number0#1) && (\number0#1 < \number0#3)}%
    {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
  \pgfmathresult
}
```

numclosedbetween

```
\dtlifnumclosedbetween{⟨num⟩}{⟨min⟩}{⟨max⟩}{⟨true part⟩}{⟨false part⟩}
```

Determines if ⟨*min*⟩ ≤ ⟨*num*⟩ ≤ ⟨*max*⟩ where all numerical arguments are in standard fixed point notation.

```
\newcommand*{\dtlifnumclosedbetween}[5]{%
```

```
    \def\@dtl@truepart{#4}%
    \def\@dtl@falsepart{#5}%
    \pgfmathifthenelse
      {(\number0#2 <= \number0#1) && (\number0#1 <= \number0#3)}
      {"\noexpand\@dtl@truepart"}{"\noexpand\@dtl@falsepart"}%
    \pgfmathresult
  }
```

3.2 Functions

\dtladd Adds two numbers using PGF math engine.

```
\newcommand*{\dtladd}[3]{%
  \pgfmathadd{#2}{#3}%
  \let#1\pgfmathresult
}
```

\dtlsub Subtracts two numbers using PGF math engine.

```
\newcommand*{\dtlsub}[3]{%
  \pgfmathsubtract{#2}{#3}%
  \let#1\pgfmathresult
}
```

\dtlmul Multiplies two numbers using PGF math engine.

```
\newcommand*{\dtlmul}[3]{%
  \pgfmathmultiply{#2}{#3}%
  \let#1\pgfmathresult
}
```

\dtldiv Divides two numbers using PGF math engine.

```
\newcommand*{\dtldiv}[3]{%
  \pgfmathdivide{#2}{#3}%
  \let#1\pgfmathresult
}
```

\dtlroot Square root using PGF math engine.

```
\newcommand*{\dtlroot}[2]{%
  \pgfmathsqrt{#2}%
  \let#1\pgfmathresult
}
```

\dtlround Rounds using PGF math engine.

```
\newcommand*{\dtlround}[3]{%
  \ifnum#3=0\relax
    \pgfmathparse{int(round(#2))}%
    \let#1\pgfmathresult
  \else
    \pgfmathparse{int(10^#3)}%
    \let\dtl@tmpshift\pgfmathresult
```

Need to be careful not to trigger the dimension too large error, so this is a bit convoluted.

```
    \pgfmathparse{int(floor(#2))}%
    \let\dtl@int@round\pgfmathresult
    \pgfmathparse{int(round((#2-\dtl@int@round) * \dtl@tmpshift))}%
```

This bit is awkward because simply dividing by multiples of 10 in pgfmath can cause rounding errors, so need to employ another method.

```
    \@dtl@tmpcount=0\relax
    \expandafter\@dtl@countdigits\pgfmathresult.\relax
    \advance\@dtl@tmpcount by -#3\relax
    \def\@dtl@intpart{}%
    \def\@dtl@fracpart{}%
    \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
    \edef\@dtl@intpart{\number\numexpr\dtl@int@round
      +\number0\@dtl@intpart}%
    \edef#1{\@dtl@intpart.\@dtl@fracpart}%
  \fi
}
```

l@gatherintfrac

```
\newcommand*{\@dtl@gatherintfrac}[1]{%
  \ifx\relax#1\relax
  \else
    \advance\@dtl@tmpcount by -1\relax
    \ifnum\@dtl@tmpcount<0\relax
      \edef\@dtl@fracpart{\@dtl@fracpart#1}%
    \else
      \edef\@dtl@intpart{\@dtl@intpart#1}%
    \fi
    \expandafter\@dtl@gatherintfrac
  \fi
}
```

\dtltrunc Truncates using PGF math engine. (Third argument is the number of digits.) This suffers from the same problems as \dtlround. Can cause dimension too large error or rounding errors.

```
\newcommand*{\dtltrunc}[3]{%
  \ifnum#3=0\relax
    \pgfmathparse{int(floor(#2))}%
    \let#1\pgfmathresult
  \else
    \pgfmathparse{int(10^#3)}%
    \let\dtl@tmpshift\pgfmathresult
```

Need to be careful not to trigger the dimension too large error, so this is a bit convoluted.

```
    \pgfmathparse{int(floor(#2))}%
    \let\dtl@int@trunc\pgfmathresult
    \pgfmathparse{int(floor((#2-\dtl@int@trunc) * \dtl@tmpshift))}%
```

This bit is awkward because simply dividing by multiples of 10 in pgfmath can cause rounding errors, so need to employ another method.

```
      \@dtl@tmpcount=0\relax
      \expandafter\@dtl@countdigits\pgfmathresult.\relax
      \advance\@dtl@tmpcount by -#3\relax
      \def\@dtl@intpart{}%
      \def\@dtl@fracpart{}%
      \expandafter\@dtl@gatherintfrac\pgfmathresult\relax
      \edef\@dtl@intpart{\number\numexpr\dtl@int@trunc
        +\number0\@dtl@intpart}%
      \edef#1{\@dtl@intpart.\@dtl@fracpart}%
    \fi
  }
```

\dtlclip There isn't a clip in pgfmath as it seems to automatically clip.

```
  \newcommand*{\dtlclip}[2]{%
   \edef#1{#2}%
  }
```

\dtlmin Minimum of two numbers using PGF math engine.

```
  \newcommand*{\dtlmin}[3]{%
    \pgfmathmin{#2}{#3}%
    \let#1\pgfmathresult
  }
```

\dtlmax Maximum of two numbers using PGF math engine.

```
  \newcommand*{\dtlmax}[3]{%
    \pgfmathmax{#2}{#3}%
    \let#1\pgfmathresult
  }
```

\dtlabs Absolute value using PGF math engine.

```
  \newcommand*{\dtlabs}[2]{%
    \pgfmathabs{#2}%
    \let#1\pgfmathresult
  }
```

\dtlneg Negative of a value using PGF math engine.

```
  \newcommand*{\dtlneg}[2]{%
    \pgfmathmul{-1}{#2}%
    \let#1\pgfmathresult
  }
```

4 datatool.sty

4.1 Package Declaration

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datatool}[2019/09/27 v2.32 (NLCT)]
```

Load required packages:

```
\RequirePackage{xkeyval}
\RequirePackage{ifthen}
\RequirePackage{xfor}
\RequirePackage{substr}
\RequirePackage{etoolbox}
```

4.2 Package Options

\@dtl@separator

The data separator character (comma by default) is stored in `\@dtl@separator`. This is the separator used in external data files, not in the LaTeX code, which always uses a comma separator.

```
\newcommand*{\@dtl@separator}{,}
```

\DTLsetseparator

`\DTLsetseparator{⟨char⟩}`

The sets `\@dtl@separator`, and constructs the relevant macros that require this character to be hardcoded into their definition.

```
\newcommand*{\DTLsetseparator}[1]{%
  \renewcommand*{\@dtl@separator}{#1}%
  \@dtl@construct@lopoffs
}
```

settabseparator

`\DTLsettabseparator` makes it easier to set a tab separator.

```
\gdef\DTLsettabseparator{%
  \catcode`\^^I12
  \DTLsetseparator{^^I}%
}
```

DTLmaketabspace

```
\gdef\DTLmaketabspace{%
  \catcode`\^^I10\relax
}
\endgroup
```

\@dtl@delimiter

The data delimiter character (double quote by default) is stored in `\@dtl@delimiter`. This is used in external data files, not in the LaTeX code.

```
\begingroup
\catcode‘\"12\relax
\gdef\@dtl@delimiter{"}
\endgroup
```

\DTLsetdelimiter

`\DTLsetdelimiter{⟨char⟩}`

This sets the delimiter.

```
\newcommand*\DTLsetdelimiter[1]{%
  \renewcommand*{\@dtl@delimiter}{#1}%
  \@dtl@construct@lopoffs
}
```

construct@lopoff

`\@dtl@construct@lopoff⟨separator char⟩⟨delimiter char⟩`

This defines

`\@dtl@lopoff⟨first element⟩⟨sep⟩⟨rest of list⟩\to⟨cmd1⟩⟨cmd2⟩`

for the current separator and delimiter.

```
\edef\@dtl@construct@lopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lopoff#1##1##2\noexpand\to##3##4{%
      \noexpand\ifx#2##1\noexpand\relax
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@qlopoff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@qlopoff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@qlopoff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\else
        \noexpand\ifstrempty{##1}%
        {\noexpand\@dtl@lop@ff#1{}##2\noexpand\to##3##4\relax}%
        {%
          \noexpand\dtl@ifsingle{##1}%
          {\noexpand\@dtl@lop@ff#1##1##2\noexpand\to##3##4\relax}%
          {\noexpand\@dtl@lop@ff#1{##1}##2\noexpand\to##3##4\relax}%
        }%
      \noexpand\fi
  }%
}
```

onstruct@qlopoff

`\@dtl@construct@qlopoff`⟨*separator char*⟩⟨*delimiter char*⟩

This constructs `\@dtl@qlopoff` to be used when the entry is surrounded by the current delimiter value.

```
\edef\@dtl@construct@qlopoff#1#2{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@qlopoff#1#2##1#2#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
```

Replace any escaped delimiters

```
      \noexpand\DTLsubstituteall{##4}{#2#2}{#2}%
      \noexpand\edef\noexpand\@dtl@dosubs{%
        \noexpand\noexpand\noexpand\DTLsubstituteall{\noexpand\noexpand##4}%
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname#2}%
        {\noexpand\expandafter\noexpand\noexpand\noexpand\csname#2\noexpand\endcsname}%
      }%
      \noexpand\@dtl@dosubs
      \noexpand\def##3{#1##2}%
  }%
}
```

construct@lop@ff

`\@dtl@construct@lop@ff`⟨*separator char*⟩

This constructs `\@dtl@lop@ff` to be used when the entry isn't surrouded by the delimiter.

```
\edef\@dtl@construct@lop@ff#1{%
  \noexpand\long
    \noexpand\def\noexpand\@dtl@lop@ff#1##1#1##2\noexpand\to##3##4{%
      \noexpand\def##4{##1}%
      \noexpand\def##3{#1##2}%
  }%
}
```

onstruct@lopoffs

`\@dtl@construct@lopoffs`

This constructs all the lopoff macros using the given separator and delimiter characters.

```
\newcommand{\@dtl@construct@lopoffs}{%
  \edef\@dtl@chars{{\@dtl@separator}{\@dtl@delimiter}}%
  \expandafter\@dtl@construct@lopoff\@dtl@chars
  \expandafter\@dtl@construct@qlopoff\@dtl@chars
  \expandafter\@dtl@construct@lop@ff\expandafter{\@dtl@separator}%
}
```

separator Define separator used in external data files.

```
\define@key{datatool.sty}{separator}{%
  \DTLsetseparator{#1}%
}
```

delimiter Define delimiter used in external data files.

```
\define@key{datatool.sty}{delimiter}{%
  \DTLsetdelimiter{#1}%
}
```

verbose

```
\define@boolkey{datatool.sty}[dtl]{verbose}[true]{}
```

math Determine whether to use fp or pgfmath for the arithmetic commands. The default is to use fp.

```
\define@choicekey{datatool.sty}{math}[\val\nr]{fp,pgfmath}{%
  \renewcommand*\@dtl@mathprocessor{#1}%
}
\providecommand*{\@dtl@mathprocessor}{fp}
```

dtl@set@options

```
\newcommand*{\@dtl@set@options}{}
```

utf8 Pass to datatool-base

```
\define@choicekey{datatool.sty}{utf8}{true,false}[true]{%
  \renewcommand*{\@dtl@set@options}{\setbool{@dtl@utf8}{#1}}%
}
```

Process package options:

```
\ProcessOptionsX
```

Set the defaults:

```
\@dtl@construct@lopoffs
```

Load base package:

```
\RequirePackage{datatool-base}
```

Set options that depend on datatool-base:

```
\@dtl@set@options
```

\DTLpar Many of the commands used by this package are short commands. This means that you can't use \par in the data. This command needs to be robust so it doesn't get expanded when written to a file. We also can't just use a synonym for \@@par because it may be used in a context where \par has a different meaning to \@@par.

```
\DeclareRobustCommand{\DTLpar}{\par}
```

4.3 Defining New Databases

As from v2.0, the internal structure of the database has changed to make it more efficient.[1] The database is now stored in a token register instead of a macro. Each row is represented as:

`\db@row@elt@w` `\db@row@id@w` *⟨row idx⟩*`\db@row@id@end@` *⟨column data⟩* `\db@row@id@w` *⟨row idx⟩*`\db@row@id@end@` `\db@row@elt@end@`

where *⟨row idx⟩* is the row index and *⟨column data⟩* is the data for each column in the row. Each column for a given row is stored as:

`\db@col@id@w` *⟨column idx⟩*`\db@col@id@end@` `\db@col@elt@w` *⟨value⟩*`\db@col@elt@end@` `\db@col@id@w` *⟨column idx⟩*`\db@col@id@end@`

where *⟨column idx⟩* is the column index and *⟨value⟩* is the entry for the given column and row.

Each row only has an associated index, but columns have a unique identifying key as well as an associated index. Columns also have an associated data type which may be: 0 (column contains strings), 1 (column contains integers), 2 (column contains real numbers), 3 (column contains currency) or *⟨empty⟩* (column contains no data). Since the key sometimes has to be expanded, a header is also available in the event that the user wants to use `\DTLdisplaydb` or `\DTLdisplaylongdb` and requires a column header that would cause problems if used as a key. The general column information is stored in a token register where each column has information stored in the form:

`\db@plist@elt@w` `\db@col@id@w` *⟨index⟩*`\db@col@id@end@` `\db@key@id@w` *⟨key⟩*`\db@key@id@end@` `\db@type@id@w` *⟨type⟩*`\db@type@id@end@` `\db@header@id@w` *⟨type⟩*`\db@header@id@end@` `\db@col@id@w` *⟨index⟩*`\db@col@id@end@` `\db@plist@elt@end@`

The column name (*⟨key⟩*) is mapped to the column index using `\dtl@ci@`*⟨db⟩*`@`*⟨key⟩* where *⟨db⟩* is the database name.

\DTLnewdb

`\DTLnewdb{⟨db name⟩}`

Initialises a database called *⟨name⟩*.

```
\newcommand*{\DTLnewdb}[1]{%
```

Check if there is already a database with this name.

```
  \DTLifdbexists{#1}%
  {%
    \PackageError{datatool}{Database ‘#1’ already exists}{}%
  }%
  {%
```

Define new database. Add information message if in verbose mode.

```
    \dtl@message{Creating database ‘#1’}%
```

Define token register used to store the contents of the database.

```
    \expandafter\newtoks\csname dtldb@#1\endcsname
```

[1] Thanks to Morten Høgholm for the suggestion.

Define token register used to store the column header information.

```
    \expandafter\newtoks\csname dtlkeys@#1\endcsname{}%
```

Define count register used to store the row count.

```
    \expandafter\newcount\csname dtlrows@#1\endcsname
```

Define count register used to store the column count.

```
    \expandafter\newcount\csname dtlcols@#1\endcsname
  }%
}
```

\DTLcleardb

\DTLcleardb{⟨*db name*⟩}

Clears the database. (Makes it empty, but still defined.)

```
\newcommand*{\DTLcleardb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \csname dtldb@#1\endcsname{}%
    \csname dtlkeys@#1\endcsname{}%
    \csname dtlrows@#1\endcsname=0\relax
    \csname dtlcols@#1\endcsname=0\relax
  }%
  {%
    \PackageError{Can't clear database '#1':
       database doesn't exist}{}{}%
  }%
}
```

\DTLdeletedb

\DTLdeletedb{⟨*db name*⟩}

Deletes a database.

```
\newcommand*{\DTLdeletedb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\let\csname dtldb@#1\endcsname\undefined
    \expandafter\let\csname dtlkeys@#1\endcsname\undefined
    \expandafter\let\csname dtlrows@#1\endcsname\undefined
    \expandafter\let\csname dtlcols@#1\endcsname\undefined
  }%
  {%
    \PackageError{Can't delete database `#1':
       database doesn't exist}{}{}%
  }%
}
```

\DTLgnewdb

\DTLgnewdb{⟨*db name*⟩}

Initialises a database called ⟨*name*⟩. (Global version.)

```
\newcommand*{\DTLgnewdb}[1]{%
```

Check if there is already a database with this name.

```
  \DTLifdbexists{#1}%
  {%
     \PackageError{datatool}{Database `#1' already exists}{}%
  }%
  {%
```

Define new database. Add information message if in verbose mode.

```
    \dtl@message{Creating database `#1'}%
```

Define token register used to store the contents of the database.

```
    \expandafter\global\expandafter\newtoks\csname dtldb@#1\endcsname
```

Define token register used to store the column header information.

```
    \expandafter\global\expandafter\newtoks\csname dtlkeys@#1\endcsname{}%
```

Define count register used to store the row count.

```
    \expandafter\global\expandafter\newcount\csname dtlrows@#1\endcsname
```

Define count register used to store the column count.

```
    \expandafter\global\expandafter\newcount\csname dtlcols@#1\endcsname
  }%
}
```

\DTLgdeletedb

\DTLgdeletedb{⟨*db name*⟩}

Deletes a database. (Global version.)

```
\newcommand*{\DTLgdeletedb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
```

```
      \expandafter\global\expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\global\expandafter\let\csname dtldb@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlkeys@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlrows@#1\endcsname\undefined
    \expandafter\global\expandafter\let\csname dtlcols@#1\endcsname\undefined
  }%
  {%
    \PackageError{Can’t delete database ‘#1’:
       database doesn’t exist}{}{}%
  }%
}
```

\DTLgcleardb

```
\DTLgcleardb{⟨db name⟩}
```

Clears the database. (Global version.)

```
\newcommand*{\DTLgcleardb}[1]{%
  \DTLifdbexists{#1}%
  {%
    \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
    {%
      \expandafter\global\expandafter\let\csname dtl@ci@#1@\@dtl@key\endcsname\undefined
    }%
    \expandafter\global\csname dtldb@#1\endcsname{}%
    \expandafter\global\csname dtlkeys@#1\endcsname{}%
    \expandafter\global\csname dtlrows@#1\endcsname=0\relax
    \expandafter\global\csname dtlcols@#1\endcsname=0\relax
  }%
  {%
    \PackageError{Can’t clear database ‘#1’:
       database doesn’t exist}{}{}%
  }%
}
```

\DTLrowcount

```
\DTLrowcount{⟨db name⟩}
```

The number of rows in the database called ⟨*db name*⟩. (Doesn’t check if database exists.)

```
\newcommand*{\DTLrowcount}[1]{%
  \expandafter\number\csname dtlrows@#1\endcsname
}
```

\DTLcolumncount

```
\DTLcolumncount{⟨db name⟩}
```

The number of columns in the database called ⟨*db name*⟩. (Doesn't check if database exists.)

```
\newcommand*{\DTLcolumncount}[1]{%
  \expandafter\number\csname dtlcols@#1\endcsname
}
```

\DTLifdbempty `\DTLifdbempty{⟨name⟩}{⟨true part⟩}{⟨false part⟩}`

Check if named database is empty (i.e. no rows have been added).

```
\newcommand{\DTLifdbempty}[3]{%
  \DTLifdbexists{#1}%
    {\@DTLifdbempty{#1}{#2}{#3}}%
    {\PackageError{Can't check if database '#1' is empty:
       database doesn't exist}{}{}}%
}
```

\@DTLifdbempty `\@sDTLifdbempty{⟨name⟩}{⟨true part⟩}{⟨false part⟩}`

Check if named existing database is empty. (No check performed to determine if the database exists.)

```
\newcommand{\@DTLifdbempty}[3]{%
  \expandafter\ifnum\csname dtlrows@#1\endcsname=0\relax
    #2%
  \else
    #3%
  \fi
}
```

\DTLnewrow `\DTLnewrow{⟨db name⟩}`

Add a new row to named database. The starred version doesn't check for the existence of the database.

```
\newcommand*{\DTLnewrow}{%
  \@ifstar\@sDTLnewrow\@DTLnewrow
}
```

\@DTLnewrow `\@DTLnewrow{⟨db name⟩}`

Add a new row to named database. (Checks for the existence of the database.)

```
\newcommand*{\@DTLnewrow}[1]{%
\DTLifdbexists{#1}%
  {\@sDTLnewrow{#1}}%
  {\PackageError{datatool}{Can't add new row to database '#1':
     database doesn't exist}{}}%
}
```

\@sDTLnewrow

\@DTLnewrow{⟨*db name*⟩}

Add a new row to named existing database. (No check performed to determine if the database exists.)

```
\newcommand*{\@sDTLnewrow}[1]{%
```

Increment row count.

```
  \global\advance\csname dtlrows@#1\endcsname by 1\relax
```

Append an empty row to the database

```
  \@dtl@toks@gput@right@cx{dtldb@#1}{%
     \noexpand\db@row@elt@w%
       \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
       \noexpand\db@row@id@end@%
       \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
       \noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
  }%
```

Display message on terminal and log file if in verbose mode.

```
  \dtl@message{New row added to database '#1'}%
}
```

\dtlcolumnnum Count register to keep track of column index.

```
\newcount\dtlcolumnnum
```

\dtlrownum Count register to keep track of row index.

```
\newcount\dtlrownum
```

\DTLifhaskey

\DTLifhaskey⟨*db name*⟩⟨*key*⟩⟨*true part*⟩⟨*false part*⟩

Checks if the named database ⟨*db name*⟩ has a column with label ⟨*key*⟩. If column exists, do ⟨*true part*⟩ otherwise do ⟨*false part*⟩. The starred version doesn't check if the named database exists.

```
\newcommand*{\DTLifhaskey}{\@ifstar\@sDTLifhaskey\@DTLifhaskey}
```

\@DTLifhaskey Unstarred version of \DTLifhaskey

```
\newcommand{\@DTLifhaskey}[4]{%
   \DTLifdbexists{#1}%
   {%
     \@sDTLifhaskey{#1}{#2}{#3}{#4}%
   }%
   {%
     \PackageError{datatool}{Database ‘#1’ doesn’t exist}{}%
   }%
}
```

\@sDTLifhaskey Starred version of \DTLifhaskey

```
\newcommand{\@sDTLifhaskey}[4]{%
  \@ifundefined{dtl@ci@#1@#2}%
  {%
```

Key not defined

```
    #4%
  }%
  {%
```

Key defined

```
    #3%
  }%
}
```

TLgetcolumnindex

```
\DTLgetcolumnindex{⟨cs⟩}{⟨db⟩}{⟨key⟩}
```

Gets index for column with label ⟨*key*⟩ from database ⟨*db*⟩ and stores in ⟨*cs*⟩ which must be a control sequence. Unstarred version checks if database and key exist, unstarred version doesn’t perform any checks.

```
\newcommand*{\DTLgetcolumnindex}{%
  \@ifstar\@sdtl@getcolumnindex\@dtl@getcolumnindex
}
```

@getcolumnindex Unstarred version of \DTLgetcolumnindex

```
\newcommand*{\@dtl@getcolumnindex}[3]{%
```

Check if database exists.

```
  \DTLifdbexists{#2}%
  {%
```

Database exists. Now check if key exists.

```
    \@sDTLifhaskey{#2}{#3}%
    {%
```

Key exists so go ahead and get column index.

```
      \@sdtl@getcolumnindex{#1}{#2}{#3}%
    }%
    {%
```

Key doesn't exists in named database.

```
      \PackageError{datatool}{Database '#2' doesn't contain
       key '#3'}{}%
    }%
  }%
  {%
```

Named database doesn't exist.

```
    \PackageError{datatool}{Database '#2' doesn't exist}{}%
  }%
}
```

@getcolumnindex

Starred version of \DTLgetcolumnindex.

```
\newcommand*{\@sdtl@getcolumnindex}[3]{%
  \expandafter\let\expandafter#1\csname dtl@ci@#2@#3\endcsname
}
```

\dtlcolumnindex

\dtlcolumnindex{⟨*db*⟩}{⟨*key*⟩}

Column index corresponding to ⟨*key*⟩ in database ⟨*db*⟩. (No check for existance of database or key.)

```
\newcommand*{\dtlcolumnindex}[2]{%
  \csname dtl@ci@#1@#2\endcsname
}
```

Lgetkeyforcolumn

\DTLgetkeyforcolumn{⟨*key cs*⟩}{⟨*db*⟩}{⟨*column index*⟩}

Gets the key associated with the given column index and stores in ⟨*key cs*⟩. Unstarred version doesn't perform checks.

```
\newcommand*{\DTLgetkeyforcolumn}{%
  \@ifstar\@sdtlgetkeyforcolumn\@dtlgetkeyforcolumn}
```

getkeyforcolumn

```
\newcommand*{\@dtlgetkeyforcolumn}[3]{%
  \DTLifdbexists{#2}%
  {%
```

Check if index is in range.

```
    \ifnum#3<1\relax
      \PackageError{datatool}{Invalid column index \number#3}{%
      Column indices start at 1}%
    \else
      \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
        \PackageError{datatool}{Index \number#3\space out of
        range for database ‘#2’}{Database ‘#2’ only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
      \else
        \@sdtlgetkeyforcolumn{#1}{#2}{#3}%
      \fi
    \fi
  }%
  {%
    \PackageError{datatool}{Database ‘#2’ doesn’t exists}{}%
  }%
}
```

lgetkeyforcolumn

\@sdtlgetkeyforcolumn{⟨*key cs*⟩}{⟨*db*⟩}{⟨*column index*⟩}

Gets the key associated with the given column index and stores in ⟨*key cs*⟩

```
\newcommand*{\@sdtlgetkeyforcolumn}[3]{%
  \edef\@dtl@dogetkeyforcolumn{\noexpand\@dtl@getkeyforcolumn
    {\noexpand#1}{#2}{\number#3}}%
  \@dtl@dogetkeyforcolumn
}
```

getkeyforcolumn

Column index must be fully expanded before use.

```
\newcommand*{\@dtl@getkeyforcolumn}[3]{%
  \def\@dtl@get@keyforcolumn##1% before stuff
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@% index
    \db@key@id@w ##2\db@key@id@end@% key
    \db@type@id@w ##3\db@type@id@end@% data type
    \db@header@id@w ##4\db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    ##5\q@nil{\def#1{##2}}%
  \edef\@dtl@tmp{\expandafter\the\csname dtlkeys@#2\endcsname}%
  \expandafter\@dtl@get@keyforcolumn\@dtl@tmp
    \db@plist@elt@w% start of block
    \db@col@id@w #3\db@col@id@end@ %index
    \db@key@id@w \@nil\db@key@id@end@% key
    \db@type@id@w \db@type@id@end@% data type
    \db@header@id@w \db@header@id@end@% header
    \db@col@id@w #3\db@col@id@end@% index
    \db@plist@elt@end@% end of block
    \q@nil
}
```

Define some commands to indicate the various data types a database may contain.

\DTLunsettype

Unknown data type. (All entries in the column are blank so the type can't be determined.)

```
\def\DTLunsettype{}
```

\DTLstringtype

Data type representing strings.

```
\def\DTLstringtype{0}
```

\DTLinttype

Data type representing integers.

```
\def\DTLinttype{1}
```

\DTLrealtype

Data type representing real numbers.

```
\def\DTLrealtype{2}
```

DTLcurrencytype

Data type representing currency.

```
\def\DTLcurrencytype{3}
```

\DTLgetdatatype

\DTLgetdatatype{⟨*cs*⟩}{⟨*db*⟩}{⟨*key*⟩}

Gets data type associated with column labelled ⟨*key*⟩ in database ⟨*db*⟩ and stores in ⟨*cs*⟩. Type may be: ⟨*empty*⟩ (unset), 0 (string), 1 (int), 2 (real), 3 (currency). Unstarred version checks if the database and key exist, starred version doesn't.

```
\newcommand*{\DTLgetdatatype}{%
  \@ifstar\@sdtlgetdatatype\@dtlgetdatatype
}
```

@dtlgetdatatype

Unstarred version of \DTLgetdatatype.

```
\newcommand*{\@dtlgetdatatype}[3]{%
```

Check if database exists.

```
  \DTLifdbexists{#2}%
  {%
```

Check if key exists in this database.

```
    \@sDTLifhaskey{#2}{#3}%
    {%
```

Get data type for this database and key.

```
      \@sdtlgetdatatype{#1}{#2}{#3}%
    }%
    {%
```

Key doesn't exist in this database.

```
      \PackageError{datatool}{Key '#3' undefined in database '#2'}{}%
    }%
  }%
  {%
```

Database doesn't exist.

```
    \PackageError{datatool}{Database '#2' doesn't exist}{}%
  }%
}
```

sdtlgetdatatype

Starred version of \DTLgetdatatype. This ensures that the key is fully expanded before begin passed to \@dtl@getdatatype.

```
\newcommand*{\@sdtlgetdatatype}[3]{%
  \edef\@dtl@dogetdata{\noexpand\@dtl@getdatatype{\noexpand#1}%
   {\expandafter\the\csname dtlkeys@#2\endcsname}%
   {\dtlcolumnindex{#2}{#3}}}%
  \@dtl@dogetdata
}
```

@dtl@getdatatype

\@dtl@getdatatype{⟨*cs*⟩}{⟨*data specs*⟩}{⟨*column index*⟩}

Column index must be expanded.

```
\newcommand*{\@dtl@getdatatype}[3]{%
  \def\@dtl@get@keydata##1% stuff before
    \db@plist@elt@w% start of key block
     \db@col@id@w #3\db@col@id@end@% column index
      \db@key@id@w ##2\db@key@id@end@% key id
      \db@type@id@w ##3\db@type@id@end@% data type
      \db@header@id@w ##4\db@header@id@end@% header
     \db@col@id@w #3\db@col@id@end@% column index
    \db@plist@elt@end@% end of key block
    ##5% stuff afterwards
    \q@nil{\def#1{##3}}%
  \@dtl@get@keydata#2\q@nil
}
```

\@dtl@getprops

\@dtl@getprops{⟨*key cs*⟩}{⟨*type cs*⟩}{⟨*header toks*⟩}{⟨*before toks*⟩}{⟨*after toks*⟩}{⟨*data specs*⟩}{⟨*column index*⟩}

Column index must be expanded.

```
\newcommand*{\@dtl@getprops}[7]{%
  \def\@dtl@get@keydata##1% stuff before
    \db@plist@elt@w% start of key block
```

```
     \db@col@id@w #7\db@col@id@end@% column index
      \db@key@id@w ##2\db@key@id@end@% key id
      \db@type@id@w ##3\db@type@id@end@% data type
      \db@header@id@w ##4\db@header@id@end@% header
     \db@col@id@w #7\db@col@id@end@% column index
    \db@plist@elt@end@% end of key block
    ##5% stuff afterwards
    \q@nil{%
      \def#1{##2}% key
      \def#2{##3}% data type
      #3={##4}% header
      #4={##1}% before stuff
      #5={##5}% after stuff
    }%
  \@dtl@get@keydata#6\q@nil
}
```

\@dtl@before

```
\newtoks\@dtl@before
```

\@dtl@after

```
\newtoks\@dtl@after
```

\@dtl@colhead

```
\newtoks\@dtl@colhead
```

\DTLaddcolumn

\DTLaddcolumn{⟨*db*⟩}{⟨*key*⟩}

Adds a column with given key to given column. No data is added to the column. The starred version doesn't check for the existence of the database.

```
\newcommand*{\DTLaddcolumn}{%
   \@ifstar\@sDTLaddcolumn\@DTLaddcolumn
}
\newcommand{\@DTLaddcolumn}[2]{%
  \DTLifdbexists{#1}%
    {\@dtl@updatekeys{#1}{#2}{}}%
    {\PackageError{datatool}{Can't add new column to database `#1':
      database doesn't exist}{}}%
}
\newcommand{\s@DTLaddcolumn}[2]{%
  \@dtl@updatekeys{#1}{#2}{}%
}
```

\@dtl@updatekeys

\@dtl@updatekeys{⟨*db*⟩}{⟨*key*⟩}{⟨*value*⟩}

Adds key to database's key list if it doesn't exist. The value is used to update the data type associated with that key. Key must be fully expanded. Doesn't check if database exists.

```
\newcommand*{\@dtl@updatekeys}[3]{%
```

Check if key already exists

```
  \@sDTLifhaskey{#1}{#2}%
  {%
```

Key exists, may need to update data type. First get the column index.

```
    \expandafter\dtlcolumnnum\expandafter
      =\dtlcolumnindex{#1}{#2}\relax
```

Get the properties for this column

```
    \edef\@dtl@dogetprops{\noexpand\@dtl@getprops
      {\noexpand\@dtl@key}{\noexpand\@dtl@type}%
      {\noexpand\@dtl@colhead}{\noexpand\@dtl@before}%
      {\noexpand\@dtl@after}{\the\csname dtlkeys@#1\endcsname}%
      {\number\dtlcolumnnum}}%
    \@dtl@dogetprops
```

Is the value empty?

```
    \ifstrempty{#3}%
    {%
```

Leave data type as it is

```
    }%
    {%
```

Make a copy of current data type

```
      \let\@dtl@oldtype\@dtl@type
```

Check the data type for this entry (stored in `\@dtl@datatype`)

```
      \@dtl@checknumerical{#3}%
```

If this column currently has no data type assigned to it then use the new type.

```
      \ifdefempty{\@dtl@type}%
      {%
        \edef\@dtl@type{\number\@dtl@datatype}%
      }%
      {%
```

This column already has an associated data type but it may need updating.

```
        \ifcase\@dtl@datatype % string
```

String overrides all other types

```
          \def\@dtl@type{0}%
        \or % int
```

All other types override int, so leave it as it is

```
        \or % real
```

Real overrides int, but not currency or string

```
          \ifnum\@dtl@type=1\relax
            \def\@dtl@type{2}%
          \fi
        \or % currency
```

Currency overrides int and real but not string

```
            \ifnum\@dtl@type>0\relax
              \def\@dtl@type{3}%
            \fi
          \fi
        }%
```

Has the data type been updated?

```
        \ifx\@dtl@oldtype\@dtl@type
```

No change needed

```
        \else
```

Update required

```
          \@dtl@toks@gconcat@middle@cx{dtlkeys@#1}%
          {\@dtl@before}%
          {%
            \noexpand\db@plist@elt@w% start of key block
              \noexpand\db@col@id@w \the\dtlcolumnnum
                \noexpand\db@col@id@end@% column index
              \noexpand\db@key@id@w #2\noexpand\db@key@id@end@% key id
              \noexpand\db@type@id@w \@dtl@type
                \noexpand\db@type@id@end@% data type
              \noexpand\db@header@id@w \the\@dtl@colhead
                \noexpand\db@header@id@end@% header
              \noexpand\db@col@id@w \the\dtlcolumnnum
                \noexpand\db@col@id@end@% column index
              \noexpand\db@plist@elt@end@% end of key block
          }%
          {\@dtl@after}%
        \fi
      }%
    }%
    {%
```

Key doesn't exist. Increment column count.

```
      \expandafter\global\expandafter\advance
        \csname dtlcols@#1\endcsname by 1\relax
      \dtlcolumnnum=\csname dtlcols@#1\endcsname\relax
```

Set column index for this key.

```
      \expandafter\xdef\csname dtl@ci@#1@#2\endcsname{%
        \number\dtlcolumnnum}%
```

Get data type for this entry (stored in `\@dtl@datatype`)

```
      \ifstrempty{#2}%
      {%
        \edef\@dtl@type{}% don't know data type yet
      }%
      {%
        \@dtl@checknumerical{#3}%
        \edef\@dtl@type{\number\@dtl@datatype}%
      }%
```

Append to property list

```
      \@dtl@toks@gput@right@cx{dtlkeys@#1}%
      {%
        \noexpand\db@plist@elt@w
        \noexpand\db@col@id@w \the\dtlcolumnnum
          \noexpand\db@col@id@end@
        \noexpand\db@key@id@w #2\noexpand\db@key@id@end@
        \noexpand\db@type@id@w \@dtl@type
          \noexpand\db@type@id@end@
        \noexpand\db@header@id@w #2\noexpand\db@header@id@end@
        \noexpand\db@col@id@w \the\dtlcolumnnum
          \noexpand\db@col@id@end@
        \noexpand\db@plist@elt@end@
      }%
    }%
  }
```

\DTLsetheader

\DTLsetheader{⟨*db*⟩}{⟨*key*⟩}{⟨*header*⟩}

Sets header for column given by ⟨*key*⟩ in database ⟨*db*⟩. Starred version doesn't check for existance of database or key.

```
  \newcommand*{\DTLsetheader}{\@ifstar\@sDTLsetheader\@DTLsetheader}
```

\@DTLsetheader Unstarred version

```
  \newcommand*{\@DTLsetheader}[3]{%
```

Check if database exists

```
    \DTLifdbexists{#1}%
    {%
```

Check if key exists.

```
      \@sDTLifhaskey{#1}{#2}%
      {%
        \@sDTLsetheader{#1}{#2}{#3}%
      }%
      {%
        \PackageError{datatool}{Database `#1' doesn't contain key
        `#2'}{}%
      }%
    }%
    {%
      \PackageError{datatool}{Database `#1' doesn't exist}{}%
```

```
  }%
}
```

\@sDTLsetheader Starred version

```
\newcommand*{\@sDTLsetheader}[3]{%
  \expandafter\dtlcolumnnum\expandafter
    =\dtlcolumnindex{#1}{#2}\relax
  \@dtl@setheaderforindex{#1}{\dtlcolumnnum}{#3}%
}
```

etheaderforindex

`\@dtl@setheaderforindex{⟨db⟩}{⟨column index⟩}{⟨header⟩}`

Sets the header for column given by ⟨*column index*⟩ in database ⟨*db*⟩. The header must be expanded.

```
\newcommand*{\@dtl@setheaderforindex}[3]{%
```

Get the properties for this column

```
  \edef\@dtl@dogetprops{\noexpand\@dtl@getprops
    {\noexpand\@dtl@key}{\noexpand\@dtl@type}%
    {\noexpand\@dtl@colhead}{\noexpand\@dtl@before}%
    {\noexpand\@dtl@after}{\the\csname dtlkeys@#1\endcsname}%
    {\number#2}}%
  \@dtl@dogetprops
```

Store the header in `\@dtl@toks`

```
  \@dtl@colhead={#3}%
```

Reconstruct property list

```
  \edef\@dtl@colnum{\number#2}\relax
  \@dtl@toks@gconcat@middle@cx{dtlkeys@#1}%
  {\@dtl@before}%
  {%
    \noexpand\db@plist@elt@w% start of block
      \noexpand\db@col@id@w \@dtl@colnum
        \noexpand\db@col@id@end@% index
      \noexpand\db@key@id@w \@dtl@key\noexpand\db@key@id@end@% key
      \noexpand\db@type@id@w \@dtl@type
        \noexpand\db@type@id@end@% data type
      \noexpand\db@header@id@w \the\@dtl@colhead
        \noexpand\db@header@id@end@% header
      \noexpand\db@col@id@w \@dtl@colnum
        \noexpand\db@col@id@end@% index
    \noexpand\db@plist@elt@end@% end of block
  }%
  {\@dtl@after}%
}
```

lexpandnewvalue Expand new value before adding to database

```
\newcommand*{\dtlexpandnewvalue}{%
  \def\@dtl@setnewvalue##1{\protected@edef\@dtl@tmp{##1}%
  \expandafter\@dtl@toks\expandafter{\@dtl@tmp}}%
}
```

oexpandnewvalue Don't expand new value before adding to database

```
\newcommand*{\dtlnoexpandnewvalue}{%
  \def\@dtl@setnewvalue##1{\@dtl@toks{##1}}%
}
```

Do this by default:

```
\dtlnoexpandnewvalue
```

\DTLnewdbentry

\DTLnewdbentry{⟨*db name*⟩}{⟨*id*⟩}{⟨*value*⟩}.

Adds an entry to the last row (adds new row if database is empty) and updates general column information if necessary. The starred version doesn't check if the database exists.

```
\newcommand{\DTLnewdbentry}{%
   \@ifstar\@sDTLnewdbentry\@DTLnewdbentry
}
```

\@DTLnewdbentry Unstarred version of \DTLnewdbentry.

```
\newcommand{\@DTLnewdbentry}[3]{%
  \DTLifdbexists{#1}%
    {\@sDTLnewdbentry{#1}{#2}{#3}}%
    {\PackageError{datatool}{Can't add new entry to database '#1':
      database doesn't exist}{}}%
}
```

@sDTLnewdbentry Starred version of \DTLnewdbentry (doesn't check if the database exists).

```
\newcommand*{\@sDTLnewdbentry}[3]{%
```

Update key list

```
  \@dtl@updatekeys{#1}{#2}{#3}%
```

Get the column index

```
  \expandafter\dtlcolumnnum\expandafter
    =\dtlcolumnindex{#1}{#2}\relax
```

Get the current row:

```
  \edef\dtl@dogetrow{\noexpand\dtlgetrow{#1}%
    {\number\csname dtlrows@#1\endcsname}}%
  \dtl@dogetrow
```

Check if this row already has an entry for the given column.

```
  \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
     {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
```

```
    }%
    \dtl@dogetentry
    \ifx\dtl@entry\dtlnovalue
```

Store the value of this entry in `\@dtl@toks`

```
      \@dtl@setnewvalue{#3}%
```

There are no entries in this row for the given column. Add this entry.

```
      \@dtl@toks@gconcat@middle@cx{dtldb@#1}%
      {\dtlbeforerow}%
      {%
```

Start of this row:

```
        \noexpand\db@row@elt@w%
```

Row ID:

```
        \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
          \noexpand\db@row@id@end@%
```

Current row so far

```
        \the\dtlcurrentrow
```

New column: Column ID

```
        \noexpand\db@col@id@w \number\dtlcolumnnum
          \noexpand\db@col@id@end@%
```

Value:

```
            \noexpand\db@col@elt@w \the\@dtl@toks
              \noexpand\db@col@elt@end@%
```

Column ID:

```
        \noexpand\db@col@id@w \number\dtlcolumnnum
          \noexpand\db@col@id@end@%
```

Row ID:

```
        \noexpand\db@row@id@w \number\csname dtlrows@#1\endcsname
          \noexpand\db@row@id@end@%
```

End of this row

```
        \noexpand\db@row@elt@end@%
      }%
```

Rest (this should be empty)

```
      {\dtlafterrow}%
```

Print information message if in verbose mode.

```
      \dtl@message{Added #2\space -> #3\space to database `#1'}%
    \else
```

There's already an entry for the given column in this row

```
      \PackageError{datatool}{Can't add entry with ID `#2' to
        current row of database `#1'}{There is already an entry with
        this ID in the current row}%
    \fi
  }
```

\DTLifdbexists

`\DTLifdbexists{⟨db name⟩}{⟨true part⟩}{⟨false part⟩}`

Checks if a data base with the given name exists.

```
\newcommand{\DTLifdbexists}[3]{%
  \@ifundefined{dtldb@#1}{#3}{#2}}
```

4.4 Accessing Data

\DTLassign

`\DTLassign{⟨db⟩}{⟨row idx⟩}{⟨assign list⟩}`

Assigns values given in ⟨*assign list*⟩ for row ⟨*row idx*⟩ in database ⟨*db*⟩. (Where ⟨*assign list*⟩ is in the same form as in `\DTLforeach`)

```
\newcommand*{\DTLassign}[3]{%
  \DTLifdbexists{#1}
  {%
```

Grouped in the event that `\dtlcurrentrow` is already in use. (Assignments in `\@dtl@assign` are global.)

```
    {%
      \dtlgetrow{#1}{#2}%
      \@dtl@assign{#3}{#1}%
    }%
  }%
  {%
    \PackageError{datatool}{Database '#1' doesn't exist}{}%
  }%
}
```

assignfirstmatch

`\DTLassignfirstmatch{⟨db⟩}{⟨col key⟩}{⟨value⟩}{⟨assign list⟩}`

Applies the assignment list to the first row that has the given value in the given column. (Value must be expanded.)

```
\newcommand*{\DTLassignfirstmatch}[4]{%
  \dtl@assignfirstmatch{#3}{#1}{#2}{#4}%
}
```

assignfirstmatch

`\xDTLassignfirstmatch{⟨db⟩}{⟨col key⟩}{⟨value⟩}{⟨assign list⟩}`

Applies the assignment list to the first row that has the given value in the given column. (Performs *one level* expansion on ⟨*value*⟩.)

```
\newcommand*{\xDTLassignfirstmatch}[4]{%
  \protected@edef\@dtl@asg@value{\expandonce{#3}}%
  \expandafter\dtl@assignfirstmatch\expandafter
   {\@dtl@asg@value}{#1}{#2}{#4}%
}
```

ssignfirstmatch Internal swaps the ordering around so the value is first. (This just makes it easier for `\xDTLassignfirstmatch`

```
\newcommand*{\dtl@assignfirstmatch}[4]{%
  \DTLifdbexists{#2}%
  {%
% Grouped in the event that \cs{dtlcurrentrow} is already in use.
% (Assignments in \cs{@dtl@assign} are global.)
%    \begin{macrocode}
    {%
```

Get row idx:

```
      \dtlgetrowindex{\dtl@asg@rowidx}{#2}{\dtlcolumnindex{#2}{#3}}{#1}%
      \ifx\dtl@asg@rowidx\dtlnovalue
        \PackageError{datatool}{No match found for
          \string\DTLassignfirstmatch{#2}{#3}{#1}{#4}}{}%
      \else
        \dtlgetrow{#2}{\dtl@asg@rowidx}%
        \@dtl@assign{#4}{#2}%
      \fi
    }%
  }%
  {%
    \PackageError{datatool}{Data base ‘#2’ doesn’t exist}{}%
  }%
}
```

\@dtl@assign `\@dtl@assign{⟨list⟩}{⟨db⟩}`

Assigns commands according to the given keys. The current row must be stored in `\dtlcurrentrow`.

```
\newcommand*{\@dtl@assign}[2]{%
  \ifstrempty{#1}{}%
  {%
    \@dtl@assigncmd#1,\@nil\@@{#2}%
  }%
}
```

\@dtl@assigncmd

\@dtl@assigncmd⟨*cmd*⟩=⟨*id*⟩\@nil

```
\def\@dtl@assigncmd#1#2=#3,#4\@@#5{%
```

Store database label. (This may already have been done so \edef is used to prevent infinite recursion.)

```
  \edef\@dtl@dbname{#5}%
```

get entry for ID given by #3 and store in #2

```
  \@sDTLifhaskey{#5}{#3}%
  {%
    \edef\@dtl@dogetentry{%
      \noexpand\dtlgetentryfromcurrentrow
        {\noexpand#1}{\dtlcolumnindex{#5}{#3}}}%
    \@dtl@dogetentry
```

Set to null if required

```
    \ifdefequal{#1}{\dtlnovalue}%
    {%
      \@@dtl@setnull{#1}{#3}%
    }%
    {}%
```

Make it global

```
    \global\let#1=#1\relax
  }%
  {%
    \PackageError{datatool}{Can't assign \string#1\space: there
     is no key '#3' in data base '#5'}{}%
```

Set to null

```
    \global\let#1\DTLstringnull
  }%
```

Recurse?

```
  \def\dtl@tmp{#4}%
  \ifx\@nnil\dtl@tmp
    \let\@dtl@next\@dtl@assigncmdnoop
  \else
    \let\@dtl@next\@dtl@assigncmd
  \fi
  \@dtl@next#4\@@{#5}%
}
```

l@assigncmdnoop

End loop

```
\def\@dtl@assigncmdnoop#1\@@#2{}
```

\@dtl@setnull `\@dtl@setnull{`⟨*cmd*⟩`}{`⟨*id*⟩`}` sets ⟨*cmd*⟩ to either `\@dtlstringnull` or `\@dtlnumbernull` depending on the data type for ⟨*id*⟩. (Database name should be stored in `\@dtl@dbname` prior to use.)

```
\newcommand*{\@dtl@setnull}[2]{%
```

Check if database given by `\@dtl@dbname` has the required key.

```
  \@sDTLifhaskey{\@dtl@dbname}{#2}%
  {%
```

Set to null

```
    \@@dtl@setnull{#1}{#2}%
  }%
  {%
```

Key not defined in database `\@dtl@dbname`.

```
    \global\let#1=\DTLstringnull
  }%
}
```

\@@dtl@setnull As above, but doesn't check if key exists

```
\newcommand*{\@@dtl@setnull}[2]{%
```

Get the data type associated with this key and store in `\@dtl@type`.

```
  \@sdtlgetdatatype{\@dtl@type}{\@dtl@dbname}{#2}%
```

Check data type.

```
  \ifnum0\@dtl@type=0\relax
```

Data type is ⟨*empty*⟩ or 0, so set to string null.

```
    \global\let#1=\DTLstringnull
  \else
```

Data type is numerical, so set to number null.

```
    \global\let#1=\DTLnumbernull
  \fi
}
```

\DTLstringnull String null value:

```
\newcommand*{\DTLstringnull}{\@dtlstringnull}
```

\@dtlstringnull String null value:

```
\newcommand*{\@dtlstringnull}{NULL}
```

\DTLnumbernull Number null value:

```
\newcommand*{\DTLnumbernull}{\@dtlnumbernull}
```

\@dtlnumbernull Number null value:

```
\newcommand*{\@dtlnumbernull}{0}
```

\DTLifnull

```
\DTLifnull{⟨command⟩}{⟨true part⟩}{⟨false part⟩}
```

Checks if ⟨*command*⟩ is null (either `\DTLstringnull` or `\DTLnumbernull`) if true, does ⟨*true part*⟩ otherwise does ⟨*false part*⟩.

```
\newcommand*{\DTLifnull}[3]{%
  \ifx#1\dtlnovalue
    #2%
  \else
    \ifx#1\DTLstringnull
      #2%
    \else
      \ifx#1\DTLnumbernull
        #2%
      \else
        #3%
      \fi
    \fi
  \fi
}
```

DTLifnullorempty

```
\DTLifnullorempty{⟨command⟩}{⟨true part⟩}{⟨false part⟩}
```

```
\newcommand*{\DTLifnullorempty}[3]{%
  \ifdefempty{#1}{#2}{\DTLifnull{#1}{#2}{#3}}%
}
```

\@dtlnovalue

```
\def\@dtlnovalue{Undefined Value}
```

\dtlnovalue

```
\def\dtlnovalue{\@dtlnovalue}
```

\DTLgetkeydata

```
\DTLgetkeydata{⟨key⟩}{⟨db⟩}{⟨col cs⟩}{⟨type cs⟩}{⟨header cs⟩}
```

Gets data for given key in database ⟨*db*⟩: the column index is stored in ⟨*col cs*⟩ and data type is stored in ⟨*type cs*⟩. The unstarred version checks for the existance of the database and key, the starred version doesn't.

```
\newcommand*{\DTLgetkeydata}{%
  \@ifstar\@sdtlgetkeydata\@dtlgetkeydata
}
```

\@dtlgetkeydata Unstarred version of \DTLgetkeydata

```
\newcommand*{\@dtlgetkeydata}[5]{%
```

Check if the database exists.

```
  \DTLifdbexists{#2}%
  {%
```

Check if the given key exists in the database.

```
    \@sDTLifhaskey{#2}{#1}%
    {%
```

Get the data.

```
      \@sdtlgetkeydata{#1}{#2}{#3}{#4}{#5}%
    }%
    {%
```

Key not defined in the given database.

```
      \PackageError{datatool}{Key ‘#1’ not defined in database
        ‘#2’}{}%
    }%
  }%
  {%
```

Database not defined.

```
    \PackageError{datatool}{Database ‘#2’ doesn’t exist}{}%
  }%
}
```

@sdtlgetkeydata \@sdtlgetkeydata{⟨*key*⟩}{⟨*db*⟩}{⟨*col cs*⟩}{⟨*type cs*⟩}{⟨*header cs*⟩} Starred verison of \DTLgetkeydata.

```
\newcommand*{\@sdtlgetkeydata}[5]{%
  \@sdtl@getcolumnindex{#3}{#2}{#1}%
  \edef\@dtl@dogetkeydata{\noexpand\@dtl@getprops
    {\noexpand\@dtl@key}{\noexpand#4}{\noexpand\@dtl@colhead}%
    {\noexpand\@dtl@before}{\noexpand\@dtl@after}%
    {\expandafter\the\csname dtlkeys@#2\endcsname}%
    {#3}}%
  \@dtl@dogetkeydata
  \edef#5{\the\@dtl@toks}%
}
```

dtl@gathervalues

\dtl@gathervalues[⟨*label*⟩]{⟨*db name*⟩}{⟨*row toks*⟩}

Stores each element of ⟨*row*⟩ in ⟨*db name*⟩ into the command \@dtl@⟨*label*⟩@⟨*key*⟩, where ⟨*key*⟩ is the key for that element, and ⟨*label*⟩ defaults to key.

```
\newcommand{\dtl@gathervalues}[3][key]{%
  \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#2}\do
  {%
```

```
    \dtlgetentryfromrow{\@dtl@tmp}{\@dtl@col}{#3}%
    \ifx\@dtl@tmp\dtlnovalue
      \@dtl@setnull{\@dtl@tmp}{\@dtl@key}%
    \fi
    \expandafter\let\csname @dtl@#1@\@dtl@key\endcsname\@dtl@tmp
  }%
}
```

l@g@gathervalues

`\dtl@g@gathervalues[⟨label⟩]{⟨db name⟩}{⟨row toks⟩}`

As above but makes global assignments

```
\newcommand{\dtl@g@gathervalues}[3][key]{%
  \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#2}\do
  {%
    \dtlgetentryfromrow{\@dtl@tmp}{\@dtl@col}{#3}%
    \ifx\@dtl@tmp\dtlnovalue
      \@dtl@setnull{\@dtl@tmp}{\@dtl@key}%
    \fi
    \expandafter\global
      \expandafter\let\csname @dtl@#1@\@dtl@key\endcsname\@dtl@tmp
  }%
}
```

\dtlcurrentrow Define token register to store current row.

```
\newtoks\dtlcurrentrow
```

\dtlbeforerow Define token register to store everything before the current row.

```
\newtoks\dtlbeforerow
```

\dtlafterrow Define token register to store everything after the current row.

```
\newtoks\dtlafterrow
```

\dtlgetrow

`\dtlgetrow{⟨db⟩}{⟨row idx⟩}`

Gets row with index ⟨*row idx*⟩ from database named ⟨*db*⟩ and stores the row in `\dtlcurrentrow`, the preceding rows in `\dtlbeforerow` and the following rows in `\dtlafterrow`. The row index, ⟨*row idx*⟩, is stored in `\dtlrownum` and the database name, ⟨*db*⟩, is stored in `\dtldbname`. This assumes that the given row exists.

```
\newcommand*{\dtlgetrow}[2]{%
  \dtlrownum=#2\relax
  \edef\dtldbname{#1}%
  \expandafter\toks@\expandafter=\csname dtldb@#1\endcsname
  \edef\@dtl@dogetrow{\noexpand\@dtlgetrow{\the\toks@}{\number#2}}%
```

```
    \@dtl@dogetrow
  }
```

tlgetrowforvalue

```
\edtlgetrowforvalue{⟨db⟩}{⟨column idx⟩}{⟨value⟩}
```

A version of `\dtlgetrowforvalue` that expands its arguments.

```
  \newcommand{\edtlgetrowforvalue}[3]{%
    \protected@edef\@dtl@dogetrowforvalue{%
      \noexpand\dtlgetrowforvalue{#1}{#2}{#3}}%
    \@dtl@dogetrowforvalue
  }
```

\DTLfetch

```
\DTLfetch{⟨db name⟩}{⟨column1 name⟩}{⟨column1 value⟩}{⟨column2 name⟩}
```

Fetches and displays the value for ⟨*column2 name*⟩ in the first row where the value of ⟨*column1 name*⟩ is ⟨*column1 value*⟩. Note that all arguments are expanded.

```
  \newcommand{\DTLfetch}[4]{%
    \edtlgetrowforvalue{#1}{\dtlcolumnindex{#1}{#2}}{#3}%
    \dtlgetentryfromcurrentrow{\dtlcurrentvalue}{\dtlcolumnindex{#1}{#4}}%
    \dtlcurrentvalue
  }
```

tlgetrowforvalue

```
\dtlgetrowforvalue{⟨db⟩}{⟨column idx⟩}{⟨value⟩}
```

Like `\dtlgetrow`, but gets the row where the entry in column ⟨*column index*⟩ matches ⟨*value*⟩. Produces an error if row not found.

```
  \newcommand*{\dtlgetrowforvalue}[3]{%
    \dtlgetrowindex{\dtl@rowidx}{#1}{#2}{#3}%
    \ifx\dtl@rowidx\dtlnovalue
      \PackageError{datatool}{No row found in database ‘#1’ for
       column ‘\number#2’ matching ‘#3’}{}%
    \else
      \dtlrownum=\dtl@rowidx\relax
      \edef\dtldbname{#1}%
      \expandafter\toks@\expandafter=\csname dtldb@#1\endcsname
      \edef\@dtl@dogetrow{\noexpand\@dtlgetrow{\the\toks@}{\dtl@rowidx}}%
      \@dtl@dogetrow
    \fi
  }
```

\@dtlgetrow

\@dtlgetrow{⟨*data specs*⟩}{⟨*row idx*⟩}

Gets the row specs from ⟨*data specs*⟩ for row with index ⟨*row idx*⟩ which must be fully expanded.

```
\newcommand*{\@dtlgetrow}[2]{%
  \def\@dtl@getrow##1% before stuff
    \db@row@elt@w% start of the row
      \db@row@id@w #2\db@row@id@end@% row id
        ##2%
      \db@row@id@w #2\db@row@id@end@% row id
    \db@row@elt@end@% end of the row
        ##3% after stuff
    \q@nil{\dtlbeforerow={##1}\dtlcurrentrow={##2}\dtlafterrow={##3}}%
  \@dtl@getrow#1\q@nil
}
```

\dtlrecombine

\dtlrecombine

Recombines database contents from \dtlbeforerow, \dtlcurrentrow and \dtlafterrow

```
\newcommand*{\dtlrecombine}{%
   \@dtl@toks@gconcat@middle@cx{dtldb@\dtldbname}%
   {\dtlbeforerow}%
   {%
```

Start of row tag

```
      \noexpand\db@row@elt@w
```

Row number

```
        \noexpand\db@row@id@w
          \number\dtlrownum
        \noexpand\db@row@id@end@
```

Current row specs:

```
         \the\dtlcurrentrow
```

Row number

```
        \noexpand\db@row@id@w
          \number\dtlrownum
        \noexpand\db@row@id@end@
```

End of row tag

```
      \noexpand\db@row@elt@end@
   }%
   {\dtlafterrow}%
}
```

mbineomitcurrent

```
\dtlrecombineomitcurrent
```

Like \dtlrecombine but omits \dtlcurrentrow

```
  \newcommand{\dtlrecombineomitcurrent}{%
```

Decrement row indices in \dtlafterrow:

```
    \dtl@decrementrows{\dtlafterrow}{\dtlrownum}
```

Reconstruct database contents by concatenating \dtlbeforerow and \dtlafterrow

```
    \csname dtldb@\dtldbname\endcsname=\dtlbeforerow
    \@dtl@toks@gput@right@cx{dtldb@\dtldbname}{\the\dtlafterrow}%
    \dtl@message{Removed row \number\dtlrownum\space in database
      '\dtldbname'}%
  }
```

\dtlsplitrow

```
\dtlsplitrow{⟨row specs⟩}{⟨col num⟩}{⟨before cs⟩}{⟨after cs⟩}
```

Splits the row around the entry given by ⟨*col num*⟩. The entries before the split are stored in ⟨*before cs*⟩ and the entries after the split are stored in ⟨*after cs*⟩. ⟨*row specs*⟩ and ⟨*col num*⟩ need to be expanded before use.

```
  \newcommand*{\dtlsplitrow}[4]{%
    \def\@dtlsplitrow##1%before stuff
      \db@col@id@w #2\db@col@id@end@% column id
        ##2% unwanted stuff
      \db@col@id@w #2\db@col@id@end@% column id
      ##3% after stuff
      \q@nil{\def#3{##1}\def#4{##3}}%
     \@dtlsplitrow#1\q@nil
  }
```

ntryincurrentrow

```
\dtlreplaceentryincurrentrow{⟨new value⟩}{⟨col num⟩}
```

Replaces entry for column ⟨*col num*⟩ in \dtlcurrentrow with ⟨*new value*⟩

```
  \newcommand*{\dtlreplaceentryincurrentrow}[2]{%
```

Split row

```
    \edef\@dtl@do@splitrow{\noexpand\dtlsplitrow
     {\the\dtlcurrentrow}%
     {\number#2}%
     {\noexpand\@dtl@before@cs}%
     {\noexpand\@dtl@after@cs}}%
    \@dtl@do@splitrow
```

Recombine with new value

```
    \toks@{#1}%
    \edef\@dtl@stuff{%
      \expandonce\@dtl@before@cs
```

Begin column index specs:

```
        \noexpand\db@col@id@w \number#2\noexpand
          \noexpand\db@col@id@end@% column id
```

New entry:

```
        \noexpand\db@col@elt@w
          \the\toks@
        \noexpand\db@col@elt@end@
```

End column index specs:

```
        \noexpand\db@col@id@w \number#2\noexpand
          \noexpand\db@col@id@end@% column id
      \expandonce\@dtl@after@cs
    }%
```

Store in `\dtlcurrentrow`

```
    \expandafter\dtlcurrentrow\expandafter{\@dtl@stuff}%
```

Update column specs

```
    \@sdtlgetkeyforcolumn{\@dtl@key}{\dtldbname}{#2}%
    \@dtl@updatekeys{\dtldbname}{\@dtl@key}{#1}%
    \dtl@message{Updated \@dtl@key\space -> #1\space in database
      `\dtldbname'}%
  }
```

ntryincurrentrow

`\dtlremoveentryincurrentrow{⟨col idx⟩}`

Removes entry for column ⟨*col idx*⟩ from `\dtlcurrentrow`.

```
  \newcommand*{\dtlremoveentryincurrentrow}[1]{%
```

Split row

```
    \edef\@dtl@do@splitrow{\noexpand\dtlsplitrow
     {\the\dtlcurrentrow}%
     {\number#1}%
     {\noexpand\@dtl@before@cs}%
     {\noexpand\@dtl@after@cs}}%
    \@dtl@do@splitrow
```

Combine row without given column:

```
    \edef\@dtl@stuff{%
      \expandonce\@dtl@before@cs
      \expandonce\@dtl@after@cs
    }%
```

Store in `\dtlcurrentrow`

```
    \expandafter\dtlcurrentrow\expandafter{\@dtl@stuff}%
    \dtl@message{Removed entry from column \number#1\space\space in database
      `\dtldbname'}%
  }
```

riesincurrentrow

```
\dtlswapentriesincurrentrow{⟨col1 num⟩}{⟨col2 num⟩}
```

Swaps columns ⟨*col1 num*⟩ and ⟨*col2 num*⟩ in `\dtlcurrentrow`

```
  \newcommand*{\dtlswapentriesincurrentrow}[2]{%
    \dtlgetentryfromcurrentrow{\@dtl@entryI}{#1}%
    \dtlgetentryfromcurrentrow{\@dtl@entryII}{#2}%
    \expandafter\dtlreplaceentryincurrentrow\expandafter
     {\@dtl@entryII}{#1}%
    \expandafter\dtlreplaceentryincurrentrow\expandafter
     {\@dtl@entryI}{#2}%
  }
```

ryfromcurrentrow

```
\dtlgetentryfromcurrentrow{⟨cs⟩}{⟨col num⟩}
```

Gets value for column ⟨*col num*⟩ from `\dtlcurrentrow` and stores in ⟨*cs*⟩. If not found, ⟨*cs*⟩ is set to `\dtlnovalue`.

```
  \newcommand*{\dtlgetentryfromcurrentrow}[2]{%
    \dtlgetentryfromrow{#1}{#2}{\dtlcurrentrow}%
  }
```

lgetentryfromrow

```
\dtlgetentryfromrow{⟨cs⟩}{⟨col num⟩}{⟨row toks⟩}
```

```
  \newcommand*{\dtlgetentryfromrow}[3]{%
    \edef\@dtl@do@getentry{\noexpand\dtl@getentryfromrow
      {\noexpand#1}{\number#2}{\the#3}}%
    \@dtl@do@getentry
  }
```

@getentryfromrow

```
\dtl@getentryfromrow{⟨cs⟩}{⟨col num⟩}{⟨row specs⟩}
```

```
  \newcommand*{\dtl@getentryfromrow}[3]{%
    \def\dtl@dogetentry##1% before stuff
```

```
      \db@col@id@w #2\db@col@id@end@% Column id
        \db@col@elt@w ##2\db@col@elt@end@% Value
      \db@col@id@w #2\db@col@id@end@% Column id
      ##3% Remaining stuff
      \q@nil{\def#1{##2}}%
    \dtl@dogetentry#3%
      \db@col@id@w #2\db@col@id@end@%
        \db@col@elt@w \@dtlnovalue\db@col@elt@end@%
      \db@col@id@w #2\db@col@id@end@%
      \q@nil
  }
```

ntrytocurrentrow

\dtlappendentrytocurrentrow{⟨*key*⟩}{⟨*value*⟩}

Appends entry to \dtlcurrentrow

```
  \newcommand*{\dtlappendentrytocurrentrow}[2]{%
```

Update information about this column (adding new column if necessary)

```
        \@dtl@updatekeys{\dtldbname}{#1}{#2}%
```

Get column index and store in \dtlcolumnnum

```
        \expandafter\dtlcolumnnum\expandafter
          =\dtlcolumnindex{\dtldbname}{#1}\relax
```

Does this row already have an entry with this key?

```
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
        }%
        \dtl@dogetentry
        \ifx\dtl@entry\dtlnovalue
```

There are no entries in this row for the given key. Expand entry value before storing.

```
          \protected@edef\@dtl@tmp{#2}%
          \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
```

Append this entry to the current row.

```
          \@dtl@toks@gput@right@cx{dtlcurrentrow}%
          {%
```

Begin column index specs:

```
            \noexpand\db@col@id@w
              \number\dtlcolumnnum
            \noexpand\db@col@id@end@
```

New entry:

```
            \noexpand\db@col@elt@w
              \the\@dtl@toks
            \noexpand\db@col@elt@end@
```

End column index specs:

```
            \noexpand\db@col@id@w
               \number\dtlcolumnnum
            \noexpand\db@col@id@end@
          }%
```

Print information to terminal and log file if in verbose mode.

```
           \dtl@message{Appended #1\space -> #2\space to database
             '\dtldbname'}%
        \else
```

There is already an entry in this row for the given key

```
          \PackageError{datatool}{Can't append entry to row:
            there is already an entry for key '#1' in this row}{}%
        \fi
  }
```

ntryincurrentrow

`\dtlupdateentryincurrentrow{⟨key⟩}{⟨value⟩}`

Appends entry to `\dtlcurrentrow` if column with given key doesn't exist, otherwise updates the value.

```
  \newcommand*{\dtlupdateentryincurrentrow}[2]{%
```

Update information about this column (adding new column if necessary)

```
        \@dtl@updatekeys{\dtldbname}{#1}{#2}%
```

Get column index and store in `\dtlcolumnnum`

```
        \expandafter\dtlcolumnnum\expandafter
          =\dtlcolumnindex{\dtldbname}{#1}\relax
```

Does this row already have an entry with this key?

```
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
        }%
        \dtl@dogetentry
        \ifx\dtl@entry\dtlnovalue
```

There are no entries in this row for the given key. Expand entry value before storing.

```
          \protected@edef\@dtl@tmp{#2}%
          \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
```

Append this entry to the current row.

```
          \@dtl@toks@gput@right@cx{dtlcurrentrow}%
          {%
```

Begin column index specs:

```
            \noexpand\db@col@id@w
              \number\dtlcolumnnum
            \noexpand\db@col@id@end@
```

New entry:

```
          \noexpand\db@col@elt@w
            \the\@dtl@toks
          \noexpand\db@col@elt@end@
```

End column index specs:

```
          \noexpand\db@col@id@w
             \number\dtlcolumnnum
          \noexpand\db@col@id@end@
        }%
```

Print information to terminal and log file if in verbose mode.

```
         \dtl@message{Appended #1\space -> #2\space to database
           `\dtldbname'}%
      \else
```

There is already an entry in this row for the given key

```
        \toks@{#2}%
        \edef\do@dtlreplaceincurrentrow{%
           \noexpand\dtlreplaceentryincurrentrow{\the\toks@}{\number\dtlcolumnnum}%
        }%
        \do@dtlreplaceincurrentrow
      \fi
}
```

\DTLgetvalue

\DTLgetvalue{⟨*cs*⟩}{⟨*db*⟩}{⟨*r*⟩}{⟨*c*⟩}

Gets the element in row ⟨*r*⟩, column ⟨*c*⟩ from database ⟨*db*⟩ and stores in ⟨*cs*⟩.

```
\newcommand*{\DTLgetvalue}[4]{%
  \edef\dtl@dogetvalue{\noexpand\dtl@getvalue{\noexpand#1}{#2}%
    {\number#3}{\number#4}}%
  \dtl@dogetvalue
}
```

\dtl@getvalue

```
\newcommand*{\dtl@getvalue}[4]{%
  \def\@dtl@getvalue ##1% stuff before row <r>
     \db@row@id@w #3\db@row@id@end@% row <r> id
        ##2% stuff in row <r> before column <c>
     \db@col@id@w #4\db@col@id@end@% column <c> id
       \db@col@elt@w ##3\db@col@elt@end@% value
     ##4% stuff after value
     \q@nil{\def#1{##3}}%
  \toks@=\csname dtldb@#2\endcsname
  \expandafter\@dtl@getvalue\the\toks@% contents of data base
     \db@row@id@w #3\db@row@id@end@%
       \db@col@id@w #4\db@col@id@end@%
```

```
         \db@col@elt@w \@dtlnovalue\db@col@elt@end@% undefined value
     \q@nil
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element at (row=#3,\space
      column=#4) in database ‘#2’}{}%
  \fi
}
```

\DTLgetlocation

```
\DTLgetlocation{⟨row cs⟩}{⟨column cs⟩}{⟨database⟩} {⟨value⟩}
```

Assigns ⟨*row cs*⟩ and ⟨*column cs*⟩ to the indices of the first entry in ⟨*database*⟩ that matches ⟨*value*⟩.

```
\newcommand*{\DTLgetlocation}[4]{%
  \def\@dtl@getlocation##1% stuff before value
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w ##2\db@col@id@end@% column id
    ##3% stuff after this column
    \db@row@id@w ##4\db@row@id@end@% row id
    ##5% stuff after row
    \q@nil{\def#1{##4}\def#2{##2}}%
  \toks@=\csname dtldb@#3\endcsname
  \expandafter\@dtl@getlocation\the\toks@% contents of data base
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w \@dtlnovalue\db@col@id@end@% undefined column id
    \db@row@id@w \@dtlnovalue\db@row@id@end@% undefined row id
    \q@nil
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element ‘#4’ in database ‘#3’}{}%
  \fi
}
```

\DTLgetrowindex

```
\DTLgetrowindex{⟨row cs⟩}{⟨database⟩}{⟨column index⟩} {⟨value⟩}
```

Assigns ⟨*row cs*⟩ to the row index of the first entry in ⟨*database*⟩ where the entry in ⟨*column index*⟩ matches ⟨*value*⟩.

```
\newcommand*{\DTLgetrowindex}[4]{%
  \toks@{#4}%
  \edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3}{\the\toks@}}%
  \dtl@dogetrowindex
  \ifx#1\dtlnovalue
    \PackageError{datatool}{There is no element ‘#4’ for column
      \number#3\space in database ‘#2’}{}%
  \fi
}
```

\dtlgetrowindex

```
\dtlgetrowindex{⟨row cs⟩}{⟨database⟩}{⟨column index⟩} {⟨value⟩}
```

As above but doesn't produce an error if not found.

```
\newcommand*{\dtlgetrowindex}[4]{%
  \toks@{#4}%
  \edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3}{\the\toks@}}%
  \dtl@dogetrowindex
}
```

\xdtlgetrowindex

```
\xdtlgetrowindex{⟨row cs⟩}{⟨database⟩}{⟨column index⟩} {⟨value⟩}
```

As above but expands the value.

```
\newcommand*{\xdtlgetrowindex}[4]{%
  \protected@edef\dtl@dogetrowindex{\noexpand\@dtlgetrowindex{\noexpand#1}{#2}{\number#3}{#4}}%
  \dtl@dogetrowindex
}
```

\@dtlgetrowindex

```
\@dtlgetrowindex{⟨row cs⟩}{⟨database⟩}{⟨column index⟩} {⟨value⟩}
```

Column index must be fully expanded.

```
\newcommand*{\@dtlgetrowindex}[4]{%
  \def\@dtl@getrowindex##1% stuff before value
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w #3\db@col@id@end@% column id
    ##2% stuff after this column
    \db@row@id@w ##3\db@row@id@end@% row id
    ##4% stuff after row
    \q@nil{\def#1{##3}}%
  \toks@=\csname dtldb@#2\endcsname
  \expandafter\@dtl@getrowindex\the\toks@% contents of data base
    \db@col@elt@w #4\db@col@elt@end@% value
    \db@col@id@w #3\db@col@id@end@% column id
    \db@row@id@w \@dtlnovalue\db@row@id@end@% undefined row id
    \q@nil
}
```

4.5 Iterating Through Databases

\@dtlforeachrow

```
\@dtlforeachrow(⟨idx cs⟩,⟨row cs⟩)\in{⟨db⟩} \do{⟨body⟩}
```

Iterates through each row in database. Assigns the current row index to ⟨*idx cs*⟩ and the row specs to ⟨*row cs*⟩

```
\long\def\@dtlforeachrow(#1,#2)\in#3\do#4{%
  \edef\dtl@tmp{\expandafter\the\csname dtldb@#3\endcsname}%
  \expandafter\@dtl@foreachrow\dtl@tmp
    \db@row@elt@w%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@@{#1}{#2}{#4}\q@nil
}
```

@dtl@foreachrow

```
\long\def\@dtl@foreachrow\db@row@elt@w%
\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@%
\db@row@elt@end@#4\@@#5#6#7\q@nil{%
```

Define control sequence given by #5

```
  \gdef#5{#1}%
```

Hide the loop body in a macro

```
  \gdef\@dtl@loopbody{#7}%
```

Increment level counter to allow for nested loops

```
  \global\advance\@dtl@foreach@level by 1\relax
```

Check if we have reached the end of the loop

```
  \ifx#5\@nnil
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop
  \else
    \gdef#6{#2}%
```

Set up the break function: Make a copy of current break function

```
    \expandafter\let
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
```

Setup break function for this level

```
    \gdef\dtlbreak{\expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop}%
```

Initialise

```
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachrow
```

Do body of loop

```
    \@dtl@loopbody
```

Restore break function

```
    \expandafter\let\expandafter\dtlbreak
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
  \fi
```

Set up what to do next.

```
  \expandafter\let\expandafter\@dtl@foreachnext
    \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
```

Decrement level counter.

```
  \global\advance\@dtl@foreach@level by -1\relax
```

Repeat loop if necessary.

```
  \@dtl@foreachnext#4\@@{#5}{#6}{#7}\q@nil
}
```

dtl@foreachnoop

```
\long\def\@dtl@foreachnoop#1\@@#2\q@nil{}
```

\dtlforeachkey

\dtlforeachkey(⟨*key cs*⟩,⟨*col cs*⟩,⟨*type cs*⟩,⟨*header cs*⟩) \in{⟨*db*⟩}\do {⟨*body*⟩}

Iterates through all the keys in database ⟨*db*⟩. In each iteration, ⟨*key cs*⟩ stores the key, ⟨*col cs*⟩ stores the column index,⟨*type cs*⟩ stores the data type and ⟨*header cs*⟩ stores the header.

```
\long\def\dtlforeachkey(#1,#2,#3,#4)\in#5\do#6{%
  \gdef\@dtl@loopbody{#6}%
  \edef\@dtl@keys{\expandafter\the\csname dtlkeys@#5\endcsname}%
  \expandafter\@dtl@foreachkey\@dtl@keys
    \db@plist@elt@w%
    \db@col@id@w -1\db@col@id@end@%
    \db@key@id@w \db@key@id@end@%
    \db@type@id@w \db@type@id@end@%
    \db@header@id@w \db@header@id@end@%
    \db@col@id@w -1\db@col@id@end@%
    \db@plist@elt@end@%
    \@@{\@dtl@updatefkcs{#1}{#2}{#3}{#4}}\q@nil
}
```

@dtl@updatefkcs

```
\newcommand*{\@dtl@updatefkcs}[8]{%
  \gdef#1{#5}%
  \gdef#2{#6}%
  \gdef#3{#7}%
  \gdef#4{#8}%
}
```

@dtl@foreachkey Sets everything globally in case it occurs in a tabular environment Loop body needs to be stored in \@dtl@loopbody. #7 indicates an update macro.

```
\long\def\@dtl@foreachkey\db@plist@elt@w%
\db@col@id@w #1\db@col@id@end@%
\db@key@id@w #2\db@key@id@end@%
\db@type@id@w #3\db@type@id@end@%
\db@header@id@w #4\db@header@id@end@%
\db@col@id@w #5\db@col@id@end@%
\db@plist@elt@end@#6\@@#7\q@nil{%
  \ifnum#1=-1\relax
```

Terminate loop

```
    \let\@dtl@foreachnext\@dtl@foreachnoop
  \else
```

Set up loop variables

```
    #7{#2}{#1}{#3}{#4}%
```

Increment level counter to allow for nested loops

```
    \global\advance\@dtl@foreach@level by 1\relax
```

Set up the break function

```
    \expandafter\let
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
      \dtlbreak
    \gdef\dtlbreak{\expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachnoop}%
```

Initialise

```
    \expandafter\global\expandafter
      \let\csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
        =\@dtl@foreachkey
```

Do body of loop

```
    \@dtl@loopbody
```

Set up what to do next

```
    \expandafter\let\expandafter\@dtl@foreachnext
      \csname @dtl@foreachnext\the\@dtl@foreach@level\endcsname
```

Restore break function

```
    \expandafter\let\expandafter\dtlbreak
      \csname @dtl@break@\the\@dtl@foreach@level\endcsname
```

Decrement level counter

```
    \global\advance\@dtl@foreach@level by -1\relax
  \fi
```

Recurse if necessary

```
  \@dtl@foreachnext#6\@@{#7}\q@nil
}
```

\dtlforcolumn

```
\dtlforcolumn{⟨cs⟩}{⟨db⟩}{⟨key⟩}{⟨body⟩}
```

Iterates through column given by ⟨*key*⟩ in database ⟨*db*⟩. ⟨*cs*⟩ is assign to the element of the column in the current iteration. Starred version doesn't check if data base exists

```
\newcommand*{\dtlforcolumn}{\@ifstar\@sdtlforcolumn\@dtlforcolumn}
```

\@dtlforcolumn

```
\newcommand{\@dtlforcolumn}[4]{%
```

Check if data base exists

```
  \DTLifdbexists{#2}%
  {%
    \@DTLifhaskey{#2}{#3}%
    {%
      \@sdtlforcolumn{#1}{#2}{#3}{#4}%
    }%
```

key not in data base

```
    {%
      \PackageError{datatool}{Database ‘#2’ doesn’t contain
        key ‘#3’}{}%
    }%
  }%
%
  {%
    \PackageError{datatool}{Database ‘#2’ doesn’t exist}{}%
  }%
}
```

\@sdtlforcolumn

```
\newcommand{\@sdtlforcolumn}[4]{%
    \toks@{#4}%
    \edef\@dtl@doforcol{\noexpand\dtl@forcolumn{\noexpand#1}%
      {\expandafter\the\csname dtldb@#2\endcsname}%
      {\dtlcolumnindex{#2}{#3}}{\the\toks@}%
    }%
    \@dtl@doforcol%
}
%    end{macrocode}
%\end{macro}
%
%\begin{macro}{\dtlforcolumnidx}
%\begin{definition}
%\cs{dtlforcolumnidx}\marg{cs}\marg{db}\marg{col num}\marg{body}
%\end{definition}
% Iterates through the column with index <col num> in database <db>.
% Starred version doesn’t check if database exists.
%\changes{2.0}{2009 February 27}{new}
```

```
%    \begin{macrocode}
\newcommand*{\dtlforcolumnidx}{%
  \@ifstar\@sdtlforcolumnidx\@dtlforcolumnidx
}
```

dtlforcolumnidx

```
\newcommand{\@dtlforcolumnidx}[4]{%
  \DTLifdbexists{#2}%
  {%
    \expandafter\ifnum\csname dtlcols@#2\endcsname<#3\relax
      \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Database `#2' only has
        \expandafter\number\csname dtlcols@#2\endcsname\space
        columns}%
    \else
      \ifnum#3<1\relax
       \PackageError{datatool}{Column index \number#3\space out of
        bounds for database `#2'}{Indices start from 1}%
      \else
        \@sdtlforcolumnidx{#1}{#2}{#3}{#4}%
      \fi
    \fi
  }%
```

data base doesn't exist

```
  {%
    \PackageError{datatool}{Database `#2' doesn't exist}{}%
  }%
}
```

dtlforcolumnidx

```
\newcommand{\@sdtlforcolumnidx}[4]{%
    \toks@{#4}%
    \edef\@dtl@doforcol{\noexpand\dtl@forcolumn{\noexpand#1}%
      {\expandafter\the\csname dtldb@#2\endcsname}%
      {\number#3}{\the\toks@}%
    }%
    \@dtl@doforcol
}
```

\dtl@forcolumn

\dtl@forcolumn{⟨*cs*⟩}{⟨*db specs*⟩}{⟨*col num*⟩}{⟨*body*⟩}

⟨*col num*⟩ needs to be fully expanded

```
\newcommand{\dtl@forcolumn}[4]{%
```

make a copy of break function

```
  \let\@dtl@oldbreak\dtlbreak
```

set up break function

```
    \def\dtlbreak{\let\@dtl@forcolnext=\@dtl@forcolnoop}%
```

define loop macro for this column

```
    \def\@dtl@forcolumn##1% before stuff
      \db@col@id@w #3\db@col@id@end@% column index
        \db@col@elt@w ##2\db@col@elt@end@% entry
      \db@col@id@w #3\db@col@id@end@% column index
      ##3% after stuff
      \q@nil{%
        \def#1{##2}% assign value to <cs>
```

check if end of loop

```
        \ifx#1\@nnil
          \let\@dtl@forcolnext=\@dtl@forcolnoop
        \else
```

do body of loop

```
          #4%
          \let\@dtl@forcolnext=\@dtl@forcolumn
        \fi
```

repeat if necessary

```
        \@dtl@forcolnext##3\q@nil
      }%
```

do loop

```
    \@dtl@forcolumn#2%
     \db@col@id@w #3\db@col@id@end@%
       \db@col@elt@w \@nil\db@col@elt@end@%
     \db@col@id@w #3\db@col@id@end@\q@nil
```

restore break function

```
    \let\dtlbreak\@dtl@oldbreak
  }
```

@dtl@forcolnoop

```
  \def\@dtl@forcolnoop#1\q@nil{}
```

dtlforeachlevel \DTLforeach can only be nested up to three levels. \dtlforeachlevel keeps track of the current level.

```
  \newcount\dtlforeachlevel
```

The counter DTLrow⟨*n*⟩ keeps track of each row of data during the ⟨*n*⟩ nested \DTLforeach. It is only incremented in the conditions (given by the optional argument) are met.

```
  \newcounter{DTLrowi}
  \newcounter{DTLrowii}
  \newcounter{DTLrowiii}
```

Keep hyperref happy

```
\newcounter{DTLrow}
\def\theHDTLrow{\arabic{DTLrow}}
\def\theHDTLrowi{\theHDTLrow.\arabic{DTLrowi}}
\def\theHDTLrowii{\theHDTLrowi.\arabic{DTLrowii}}
\def\theHDTLrowiii{\theHDTLrowii.\arabic{DTLrowiii}}
```

```
\newcount\dtl@rowi
\newcount\dtl@rowii
\newcount\dtl@rowiii
```

```
\newtoks\@dtl@curi
\newtoks\@dtl@previ
\newtoks\@dtl@nexti
\newtoks\@dtl@curii
\newtoks\@dtl@previi
\newtoks\@dtl@nextii
\newtoks\@dtl@curiii
\newtoks\@dtl@previii
\newtoks\@dtl@nextiii
```

\DTLsaverowcount

`\DTLsavelastrowcount{⟨cmd⟩}`

Stores the maximum row count for the last `\DTLforeach`.

```
\newcommand*{\DTLsavelastrowcount}[1]{%
\ifnum\dtlforeachlevel>2\relax
  \def#1{0}%
\else
  \ifnum\dtlforeachlevel<0\relax
    \def#1{0}%
  \else
    \@dtl@tmpcount=\dtlforeachlevel
    \advance\@dtl@tmpcount by 1\relax
    \edef#1{\expandafter\number
      \csname c@DTLrow\romannumeral\@dtl@tmpcount\endcsname}%
  \fi
\fi}
```

DTLenvforeach

Environment form of `\DTLforeach` (contents are gathered, so verbatim can't be used).

```
\newenvironment{DTLenvforeach}[3][\boolean{true}]%
{%
  \def\@dtlenvforeach@args{[#1]{#2}{#3}}%
  \long@collect@body\@do@dtlenvforeach
}%
{}
\newcommand{\@do@dtlenvforeach}[1]{%
  \expandafter\@DTLforeach\@dtlenvforeach@args{#1}%
}
```

DTLenvforeach* Environment form of \DTLforeach* (contents are gathered, so verbatim can't be used).

```
\newenvironment{DTLenvforeach*}[3][\boolean{true}]%
{%
  \def\s@dtlenvforeach@args{[#1]{#2}{#3}}%
  \long@collect@body\@do@sdtlenvforeach
}%
{}
\newcommand{\@do@sdtlenvforeach}[1]{%
  \expandafter\@sDTLforeach\s@dtlenvforeach@args{#1}%
}
```

\DTLforeach

```
\DTLforeach[⟨conditions⟩]{⟨db name⟩}{⟨values⟩}{⟨text⟩}
```

For each row of data in the database given by ⟨*db name*⟩, do ⟨*text*⟩, if the specified conditions are satisfied. The argument {⟨*values*⟩} is a comma separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. At the start of each row, each of the commands in this list are set to the value of the entry with the corresponding key ⟨*key*⟩. (\gdef is used to ensure \DTLforeach works in a tabular environment.) The database may be edited in the unstarred version, in the starred version the database is read only.

```
\newcommand*{\DTLforeach}{\@ifstar\@sDTLforeach\@DTLforeach}
```

\@DTLforeach \@DTLforeach is the unstarred version of \DTLforeach. The database is reconstructed to allow for rows to be edited. Use the starred version for faster access.

```
\newcommand{\@DTLforeach}[4][\boolean{true}]{%
```

Check database exists

```
  \DTLifdbexists{#2}%
  {%
```

Keep hyperref happy

```
    \refstepcounter{DTLrow}%
```

Make it global (so that it works in tabular environment)

```
    \global\c@DTLrow=\c@DTLrow\relax
```

Store database name

```
    \xdef\@dtl@dbname{#2}%
```

Increment level and check not exceeded 3

```
    \global\advance\dtlforeachlevel by 1\relax
    \ifnum\dtlforeachlevel>3\relax
      \PackageError{datatool}{\string\DTLforeach\space nested too
        deeply}{Only 3 levels are allowed}%
    \else
      \@DTLifdbempty{#2}%
```

Do nothing if database is empty

```
      {}%
      {%
```

Set level dependent information (needs to be global to ensure it works in the tabular environment). Row counter:

```
\expandafter\global
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
    = 0\relax
```

Store previous value of \DTLiffirstrow

```
\expandafter\global\expandafter\let%
  \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
  \DTLiffirstrow
```

Define current \DTLiffirstrow

```
\gdef\DTLiffirstrow##1##2{%
  \expandafter\ifnum
   \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
   =1 %space intended
    ##1%
  \else
    ##2%
  \fi}%
```

Store previous value of \DTLiflastrow

```
\expandafter\global\expandafter\let%
  \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
  \DTLiflastrow
```

Define current \DTLiflastrow

```
\gdef\DTLiflastrow##1##2{%
  \expandafter\ifnum
   \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
   =\csname dtlrows@#2\endcsname
    ##1%
  \else
    ##2%
  \fi}%
```

Store previous value of \DTLifoddrow

```
\expandafter\global\expandafter\let%
  \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
  \DTLifoddrow
```

Define current \DTLifoddrow

```
\gdef\DTLifoddrow##1##2{%
  \expandafter\ifodd
   \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
    ##1%
  \else
    ##2%
  \fi}%
```

Store data base name for current level

```
\expandafter\global\expandafter\let
```

```
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
        =\@dtl@dbname
```

Mark it as not read only

```
    \expandafter\global\expandafter\let
      \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname
        = 0\relax
```

Loop through each row. Loop counter given by `\dtl@row`⟨*level*⟩

```
    \dtlgforint
      \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
      =1\to\csname dtlrows@#2\endcsname\step1\do
    {%
```

Get current row from the data base

```
      \@dtl@tmpcount=
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
      \edef\dtl@dogetrow{\noexpand\dtlgetrow{#2}%
        {\number\@dtl@tmpcount}}%
      \dtl@dogetrow
```

Store the current row for this level

```
      \expandafter\global
        \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
          = \dtlcurrentrow
```

Store the previous rows for this level

```
      \expandafter\global
        \csname @dtl@prev\romannumeral\dtlforeachlevel\endcsname
          = \dtlbeforerow
```

Store the subsequent rows for this level

```
      \expandafter\global
        \csname @dtl@next\romannumeral\dtlforeachlevel\endcsname
          = \dtlafterrow
```

Assign commands to the required entries

```
      \ifblank{#3}{}{\@dtl@assign{#3}{#2}}%
```

Do the main body of text if condition is satisfied

```
      \ifthenelse{#1}%
      {%
```

Increment user row counter

```
        \refstepcounter{DTLrow\romannumeral\dtlforeachlevel}%
        \expandafter\edef\expandafter\DTLcurrentindex%
          \expandafter{%
             \arabic{DTLrow\romannumeral\dtlforeachlevel}}%
        #4%
```

Has this row been marked for deletion?

```
        \edef\@dtl@tmp{\expandafter\the
          \csname @dtl@cur\romannumeral
        \dtlforeachlevel\endcsname}%
    \ifx\@dtl@tmp\@nnil
```

Row needs to be deleted Decrement row indices for rows with a higher index than this one

```
      \expandafter\dtl@decrementrows\expandafter
         {\csname @dtl@prev\romannumeral
            \dtlforeachlevel\endcsname
         }{\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname}%
      \expandafter\dtl@decrementrows\expandafter
         {\csname @dtl@next\romannumeral
            \dtlforeachlevel\endcsname
         }{\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname}%
```

Reconstruct data base without this row

```
      \edef\@dtl@tmp{%
        \expandafter\the
          \csname @dtl@prev\romannumeral
            \dtlforeachlevel\endcsname
        \expandafter\the
          \csname @dtl@next\romannumeral
             \dtlforeachlevel\endcsname
        }%
      \expandafter\global\expandafter
         \csname dtldb@#2\endcsname\expandafter{\@dtl@tmp}%
```

Decrement the row count for this database:

```
      \expandafter\global\expandafter
        \advance\csname dtlrows@#2\endcsname by -1\relax
```

Decrement the counter for this loop

```
      \expandafter\global\expandafter
        \advance\csname dtl@row\romannumeral
           \dtlforeachlevel\endcsname by -1\relax
    \else
```

Reconstruct data base

```
      \@dtl@before=\csname @dtl@prev\romannumeral
        \dtlforeachlevel\endcsname
      \@dtl@after=\csname @dtl@next\romannumeral
        \dtlforeachlevel\endcsname
      \@dtl@toks@gconcat@middle@cx{dtldb@#2}%
      {\@dtl@before}%
      {%
```

This row

```
        \noexpand\db@row@elt@w%
        \noexpand\db@row@id@w \expandafter\number
          \csname dtl@row\romannumeral
            \dtlforeachlevel\endcsname
        \noexpand\db@row@id@end@%
```

```
                  \expandafter\the
                    \csname @dtl@cur\romannumeral
                      \dtlforeachlevel\endcsname
                  \noexpand\db@row@id@w \expandafter\number
                    \csname dtl@row\romannumeral
                      \dtlforeachlevel\endcsname
                  \noexpand\db@row@id@end@%
                  \noexpand\db@row@elt@end@%
                  }%
                {\@dtl@after}%
              \fi
            }%
```

Condition not met so ignore

```
            {}%
           }%
```

Restore previous value of \DTLiffirstrow

```
          \expandafter\global\expandafter\let\expandafter\DTLiffirstrow
            \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
```

Restore previous value of \DTLiflastrow

```
          \expandafter\global\expandafter\let\expandafter\DTLiflastrow
            \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
```

Restore previous value of \DTLifoddrow

```
          \expandafter\global\expandafter\let\expandafter\DTLifoddrow
            \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
        }%
      \fi
```

Decrement level

```
      \global\advance\dtlforeachlevel by -1\relax
    }%
```

else part (data base doesn't exist):

```
    {%
      \PackageError{datatool}{Database ‘#2’ doesn’t exist}{}%
    }%
  }
```

\@sDTLforeach \@sDTLforeach is the starred version of \DTLforeach. The database rows can't be edited.

```
  \newcommand{\@sDTLforeach}[4][\boolean{true}]{%
```

Check database exists

```
    \DTLifdbexists{#2}%
    {%
```

Keep hyperref happy

```
      \refstepcounter{DTLrow}%
```

Make it global (so that it works in tabular environment)

```
      \global\c@DTLrow=\c@DTLrow
```

Store database name.

```
\xdef\@dtl@dbname{#2}%
```

Increment level and check not exceeded 3

```
\global\advance\dtlforeachlevel by 1\relax
\ifnum\dtlforeachlevel>3\relax
  \PackageError{datatool}{\string\DTLforeach\space nested too
    deeply}{Only 3 levels are allowed}%
\else
   \@DTLifdbempty{#2}%
```

Do nothing if database is empty

```
   {}%
   {%
```

Set level dependent information (needs to be global to ensure it works in the tabular environment). Row counter:

```
    \expandafter\global
      \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
        = 0\relax
```

Store previous value of `\DTLiffirstrow`

```
    \expandafter\global\expandafter\let%
      \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
      \DTLiffirstrow
```

Define current `\DTLiffirstrow`

```
    \gdef\DTLiffirstrow##1##2{%
      \expandafter\ifnum
       \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
       =1 % space intended
        ##1%
      \else
        ##2%
      \fi}%
```

Store previous value of `\DTLiflastrow`

```
    \expandafter\global\expandafter\let%
      \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
      \DTLiflastrow
```

Define current `\DTLiflastrow`

```
    \gdef\DTLiflastrow##1##2{%
      \expandafter\ifnum
       \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
       =\csname dtlrows@#2\endcsname
        ##1%
      \else
        ##2%
      \fi}%
```

Store previous value of \DTLifoddrow

```
\expandafter\global\expandafter\let%
  \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
  \DTLifoddrow
```

Define current \DTLifoddrow

```
\gdef\DTLifoddrow##1##2{%
  \expandafter\ifodd
   \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname
    ##1%
  \else
    ##2%
  \fi}%
```

Store data base name for current level

```
\expandafter\gdef\csname @dtl@dbname@\romannumeral
  \dtlforeachlevel\endcsname{#2}%
```

Mark it as read only

```
\expandafter\global\expandafter\let
  \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname
    = 1\relax
```

Iterate through each row.

```
\@dtlforeachrow(\dtl@thisidx,\dtl@thisrow)\in{#2}\do%
{%
```

Assign row number (not sure if this is needed here)

```
  \csname dtl@row\romannumeral\dtlforeachlevel\endcsname
    = \dtl@thisidx\relax
```

Store the current row specs for this level

```
  \expandafter\global
    \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
      = \expandafter{\dtl@thisrow}%
```

Assign commands to the required entries

```
  \ifblank{#3}{}
  {%
```

Need to set \dtlcurrentrow for \@dtl@assign

```
    \dtlcurrentrow=\expandafter{\dtl@thisrow}%
    \@dtl@assign{#3}{#2}%
  }%
```

Do the main body of text if condition is satisfied

```
  \ifthenelse{#1}%
  {%
```

Increment user row counter

```
    \refstepcounter{DTLrow\romannumeral\dtlforeachlevel}%
    \expandafter\edef\expandafter\DTLcurrentindex%
      \expandafter{%
```

```
                 \arabic{DTLrow\romannumeral\dtlforeachlevel}}%
            #4%
          }%
```

Condition not met so ignore

```
          {}%
         }%
```

Restore previous value of \DTLiffirstrow

```
        \expandafter\global\expandafter\let\expandafter\DTLiffirstrow
          \csname @dtl@iffirstrow\the\dtlforeachlevel\endcsname
```

Restore previous value of \DTLiflastrow

```
        \expandafter\global\expandafter\let\expandafter\DTLiflastrow
          \csname @dtl@iflastrow\the\dtlforeachlevel\endcsname
```

Restore previous value of \DTLifoddrow

```
        \expandafter\global\expandafter\let\expandafter\DTLifoddrow
          \csname @dtl@ifoddrow\the\dtlforeachlevel\endcsname
      }%
    \fi
```

Decrement level

```
    \global\advance\dtlforeachlevel by -1\relax
  }%
```

else part (data base doesn't exist):

```
  {%
    \PackageError{datatool}{Database ‘#2’ doesn’t exist}{}%
  }%
}
```

\@dtlifreadonly

\@dtlifreadonly{⟨*true part*⟩}{⟨*false part*⟩}

Checks if current loop level is read only

```
\newcommand*{\@dtlifreadonly}[2]{%
  \expandafter\ifx
    \csname @dtl@ro@\romannumeral\dtlforeachlevel\endcsname1\relax
```

Read only

```
    #1%
  \else
```

Not read only

```
    #2%
  \fi
}
```

\DTLappendtorow

```
\DTLappendtorow{⟨key⟩}{⟨value⟩}
```

Appends entry to current row. (The current row is given by `\@dtl@cur⟨n⟩` where ⟨*n*⟩ is roman numeral value of `\dtlforeachlevel`. One level expansion is applied to ⟨*value*⟩.

```
\newcommand*{\DTLappendtorow}[2]{%
  \ifnum\dtlforeachlevel=0\relax
    \PackageError{datatool}{\string\DTLappendrow\space can only be
      used inside \string\DTLforeach}{}%
  \else
```

Set `\@dtl@thisdb` to the current database name:

```
    \expandafter\let\expandafter\@dtl@thisdb
      \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Check this isn’t in `\DTLforeach*`

```
    \@dtlifreadonly
    {%
      \PackageError{datatool}{\string\DTLappendtorow\space can't
       be used inside \DTLforeach*}{The starred version of
       \string\DTLforeach\space is read only}%
    }%
    {%
```

Store current row number in `\dtlrownum`

```
      \dtlrownum=
        \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
```

Update information about this column (adding new column if necessary)

```
      \@dtl@updatekeys{\@dtl@thisdb}{#1}{#2}%
```

Get column index and store in `\dtlcolumnnum`

```
      \expandafter\dtlcolumnnum\expandafter
        =\dtlcolumnindex{\@dtl@thisdb}{#1}\relax
```

Set `\dtlcurrentrow` to the current row

```
      \dtlcurrentrow =
        \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row already have an entry with this key?

```
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
        {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
      }%
      \dtl@dogetentry
      \ifx\dtl@entry\dtlnovalue
```

There are no entries in this row for the given key. Expand entry value before storing.

```
        \protected@edef\@dtl@tmp{#2}%
        \expandafter\@dtl@toks\expandafter{\@dtl@tmp}%
```

Append this entry to the current row.

```
          \@dtl@toks@gput@right@cx{@dtl@cur\romannumeral\dtlforeachlevel}%
          {%
            \noexpand\db@col@id@w \number\dtlcolumnnum
              \noexpand\db@col@id@end@
            \noexpand\db@col@elt@w \the\@dtl@toks
              \noexpand\db@col@elt@end@
            \noexpand\db@col@id@w \number\dtlcolumnnum
              \noexpand\db@col@id@end@
          }%
```

Print information to terminal and log file if in verbose mode.

```
           \dtl@message{Appended #1\space -> #2\space to database
             '\@dtl@thisdb'}%
        \else
```

There is already an entry in this row for the given key

```
          \PackageError{datatool}{Can't append entry to row:
            there is already an entry for key '#1' in this row}{}%
        \fi
      }%
    \fi
  }
```

moveentryfromrow

\DTLremoveentryfromrow{⟨*key*⟩}

Removes entry given by ⟨*key*⟩ from current row. (The current row is given by \@dtl@cur⟨*n*⟩ where ⟨*n*⟩ is roman numeral value of \dtlforeachlevel.

```
  \newcommand*{\DTLremoveentryfromrow}[1]{%
    \ifnum\dtlforeachlevel=0\relax
      \PackageError{datatool}{\string\DTLremoventryfromrow\space
        can only be used inside \string\DTLforeach}{}%
    \else
```

Set \@dtl@thisdb to the current database name:

```
      \expandafter\let\expandafter\@dtl@thisdb
        \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Check this isn't in \DTLforeach*

```
      \@dtlifreadonly
      {%
        \PackageError{datatool}{\string\DTLremoveentryfromrow\space
          can't be used inside \string\DTLforeach*}{The starred
          version of \string\DTLforeach\space is read only}%
      }%
      {%
```

Store current row number in \dtlrownum

```
    \dtlrownum=
      \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
```

Is there a column corresponding to this key?

```
    \@DTLifhaskey{\@dtl@thisdb}{#1}%
    {%
```

There exists a column for this key, so get the index:

```
      \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}\relax
      \dtlcolumnnum=\thiscol\relax
```

Set \dtlcurrentrow to the current row

```
      \dtlcurrentrow =
        \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row have an entry with this key?

```
      \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
        {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
      }%
      \dtl@dogetentry
      \ifx\dtl@entry\dtlnovalue
```

This row doesn't contain an entry with this key

```
        \PackageError{datatool}{Can't remove entry given by `#1'
           from current row in database `\@dtl@thisdb': no such
           entry}{The current row doesn't contain an entry for
           key `#1'}%
      \else
```

Split the current row around the unwanted entry

```
        \edef\@dtl@dosplitrow{%
          \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
            {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
            {\noexpand\dtl@post}%
        }%
        \@dtl@dosplitrow
```

Reconstruct row without unwanted entry

```
        \expandafter\@dtl@toks\expandafter{\dtl@pre}%
        \expandafter\toks@\expandafter{\dtl@post}%
        \edef\@dtl@tmp{\the\@dtl@toks \the\toks@}%
        \dtlcurrentrow=\expandafter{\@dtl@tmp}%
        \expandafter\global
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
            = \dtlcurrentrow
       \dtl@message{Removed entry given by #1\space from current
         row of database `\@dtl@thisdb'}%
      \fi
    }%
    {%
      \PackageError{datatool}{Can't remove entry given by
```

```
            ‘#1’ - no such key exists}{}%
        }%
      }%
    \fi
  }
```

placeentryforrow

```
\DTLreplaceentryforrow{⟨key⟩}{⟨value⟩}
```

Replaces entry given by ⟨*key*⟩ in current row with ⟨*value*⟩. (The current row is given by the token register `\@dtl@cur`⟨*n*⟩ where ⟨*n*⟩ is roman numeral value of `\dtlforeachlevel`.

```
  \newcommand*{\DTLreplaceentryforrow}[2]{%
    \ifnum\dtlforeachlevel=0\relax
      \PackageError{datatool}{\string\DTLreplaceentryforrow\space
        can only be used inside \string\DTLforeach}{}%
    \else
```

Set `\@dtl@thisdb` to the current database name:

```
      \expandafter\let\expandafter\@dtl@thisdb
        \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Check this isn’t in `\DTLforeach*`

```
      \@dtlifreadonly
      {%
        \PackageError{datatool}{\string\DTLreplaceentryforrow\space
          can't be used inside \string\DTLforeach*}{The starred version
          of \string\DTLforeach\space is read only}%
      }%
      {%
```

Store current row number in `\dtlrownum`

```
        \dtlrownum=
          \csname dtl@row\romannumeral\dtlforeachlevel\endcsname\relax
```

Is there a column corresponding to this key?

```
        \@DTLifhaskey{\@dtl@thisdb}{#1}%
        {%
```

There exists a column for this key, so get the index:

```
          \@dtl@getcolumnindex{\thiscol}{\@dtl@thisdb}{#1}\relax
          \dtlcolumnnum=\thiscol\relax
```

Set `\dtlcurrentrow` to the current row

```
          \dtlcurrentrow =
            \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Does this row have an entry with this key?

```
          \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
            {\noexpand\dtl@entry}{\number\dtlcolumnnum}%
          }%
          \dtl@dogetentry
          \ifx\dtl@entry\dtlnovalue
```

This row doesn’t contain an entry with this key

```
            \PackageError{datatool}{Can’t replace entry given by ‘#1’
               from current row in database ‘\@dtl@thisdb’: no such
               entry}{The current row doesn’t contain an entry for
               key ‘#1’}%
          \else
```

Split the current row around the requested entry

```
            \edef\@dtl@dosplitrow{%
              \noexpand\dtlsplitrow{\the\dtlcurrentrow}%
                {\number\dtlcolumnnum}{\noexpand\dtl@pre}%
                {\noexpand\dtl@post}%
            }%
            \@dtl@dosplitrow
```

Reconstruct row with new value (given by #2).

```
            \protected@edef\@dtl@tmp{#2}%
            \expandafter\@dtl@toks\expandafter{\@dtl@tmp}% new value
            \expandafter\@dtl@before\expandafter{\dtl@pre}%
            \expandafter\@dtl@after\expandafter{\dtl@post}%
            \@dtl@toks@gconcat@middle@cx
              {@dtl@cur\romannumeral\dtlforeachlevel}%
              {\@dtl@before}%
              {%
                \noexpand\db@col@id@w \number\dtlcolumnnum
                  \noexpand\db@col@id@end@%
                \noexpand\db@col@elt@w \the\@dtl@toks
                  \noexpand\db@col@elt@end@%
                \noexpand\db@col@id@w \number\dtlcolumnnum
                  \noexpand\db@col@id@end@%
              }%
              {\@dtl@after}%
```

Print information to terminal and log file if in verbose mode.

```
           \dtl@message{Updated #1\space -> #2\space in database
             ‘\@dtl@thisdb’}%
          \fi
        }%
        {%
```

There doesn’t exist a column for this key.

```
           \PackageError{datatool}{Can’t replace key ‘#1’ - no such
             key in database ‘\@dtl@thisdb’}{}%
        }%
      }%
    \fi
  }
```

removecurrentrow

```
\DTLremovecurrentrow
```

Removes current row. This just sets the current row to empty

```
  \newcommand*{\DTLremovecurrentrow}{%
    \ifnum\dtlforeachlevel=0\relax
      \PackageError{datatool}{\string\DTLremovecurrentrow\space can
        only be used inside \string\DTLforeach}{}%
    \else
```

Set `\@dtl@thisdb` to the current database name:

```
      \expandafter\let\expandafter\@dtl@thisdb
        \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Check this isn't in `\DTLforeach*`

```
      \@dtlifreadonly
      {%
        \PackageError{datatool}{\string\DTLreplaceentryforrow\space
          can't be used inside \string\DTLforeach*}{The starred version
          of \string\DTLforeach\space is read only}%
      }%
      {%
```

Set the current row to `\@nil` (`\DTLforeach` needs to check for this)

```
        \expandafter\global
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
            ={\@nil}%
      }%
    \fi
  }
```

TLaddentryforrow

```
\DTLaddentryforrow{⟨db name⟩}{⟨assign list⟩}{⟨condition⟩}{⟨key⟩}{⟨value⟩}
```

Adds the entry with key given by ⟨*key*⟩ and value given by ⟨*value*⟩ to the first row in the database ⟨*db name*⟩ which satisfies the condition given by ⟨*condition*⟩. The ⟨*assign list*⟩ is the same as for `\DTLforeach` and may be used to set the values which are to be tested in ⟨*condition*⟩.

```
  \newcommand{\DTLaddentryforrow}[5]{%
```

Iterate through the data base until condition is met

```
    \DTLifdbexists{#1}%
    {%
      \def\@dtl@notdone{\PackageError{datatool}{Unable to add entry
        given by key '#4': condition not met for any row in database
        '#1'}{}}%
```

Iterate through each row

```
      \DTLforeach[#3]{#1}{#2}%
      {%
```

add entry to this row

```
        \DTLappendtorow{#4}{#5}%
```

disable error message

```
        \let\@dtl@notdone\relax
```

break out of loop

```
        \dtlbreak
      }%
      \@dtl@notdone
    }%
    {%
      \PackageError{datatool}{Unable to add entry given by key '#4':
        database '#1' doesn't exist}{}%
    }%
  }
```

Lforeachkeyinrow

```
\DTLforeachkeyinrow{⟨cmd⟩}{⟨text⟩}
```

Iterates through each key in the current row of `\DTLforeach`, and does ⟨*text*⟩.

```
  \newcommand*{\DTLforeachkeyinrow}[2]{%
    \ifnum\dtlforeachlevel=0\relax
      \PackageError{datatool}{\string\DTLforeachkeyinrow\space can only
       be used inside \string\DTLforeach}{}%
    \else
```

Set `\@dtl@thisdb` to the current database name:

```
      \expandafter\let\expandafter\@dtl@thisdb
        \csname @dtl@dbname@\romannumeral\dtlforeachlevel\endcsname
```

Iterate through key list

```
      \dtlforeachkey(\dtlkey,\dtlcol,\dtltype,\dtlheader)\in
        \@dtl@thisdb\do{%
```

store row in `\dtlcurrentrow` (This may get nested so need to do it here instead of outside this loop in case ⟨*text*⟩ changes it.)

```
        \dtlcurrentrow =
          \csname @dtl@cur\romannumeral\dtlforeachlevel\endcsname
```

Get the value for this key and store in #1

```
        \edef\dtl@dogetentry{\noexpand\dtlgetentryfromcurrentrow
          {\noexpand#1}{\dtlcol}}%
        \dtl@dogetentry
```

Check if null

```
      \ifx#1\dtlnovalue
        \ifnum0\dtltype=0\relax
```

Data type is ⟨*empty*⟩ or 0, so set to string null.

```
          \let#1=\@dtlstringnull
        \else
```

Data type is numerical, so set to number null.

```
          \let#1=\@dtlnumbernull
        \fi
      \fi
```

Make #1 global in case this is in a tabular environment (or something similar)

```
      \global\let#1#1%
```

Store loop body so that any scoping commands (such as &) don't cause a problem for `\ifx`

```
      \def\@dtl@loop@body{#2}%
      \@dtl@loop@body
    }%
  \fi
}
```

4.6 DTLforeach Conditionals

The following conditionals are only meant to be used within `\DTLforeach` as they depend on the counter `DTLrow`⟨*n*⟩.

`\DTLiffirstrow`

`\DTLiffirstrow{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Test if the current row is the first row. (This takes ⟨*condition*⟩, the optional argument of `\DTLforeach`, into account, so it may not correspond to row 1 of the database.) Can only be used in `\DTLforeachrow`.

```
\newcommand{\DTLiffirstrow}[2]{%
  \PackageError{datatool}{\string\DTLiffirstrow\space can only
  be used inside \string\DTLforeach}{}%
}
```

`\DTLiflastrow`

`\DTLiflastrow{`⟨*true part*⟩`}{`⟨*false part*⟩`}`

Checks if the current row is the last row of the database. It doesn't take the condition (the optional argument of `\DTLforeach`) into account, so its possible it may never do ⟨*true part*⟩, as the last row of the database may not meet the condition. It is therefore not very useful and is confusing since it behaves differently to `\DTLiffirstrow` which does take the condition into account, so I have removed its description from the main part of the manual. If you need to use the optional argument of \DTLforeach, you will first have to iterate through the database to count up the number of rows which meet the condition, and then do another pass, checking if the current row has reached that number.

```
\newcommand{\DTLiflastrow}[2]{%
  \PackageError{datatool}{\string\DTLiflastrow\space can only
  be used inside \string\DTLforeach}{}%
}
```

\DTLifoddrow

```
\DTLifoddrow{⟨true part⟩}{⟨false part⟩}
```

Determines whether the current row is odd (takes the optional argument of \DTLforeach into account.)

```
\newcommand{\DTLifoddrow}[2]{%
  \PackageError{datatool}{\string\DTLifoddrow\space can only
  be used inside \string\DTLforeach}{}%
}
```

4.7 Displaying Database

This section defines commands to display the entire database in a tabular or longtable environment.

\dtlbetweencols This specifies what to put between the column alignment specifiers.

```
\newcommand*{\dtlbetweencols}{}
```

\dtlbeforecols This specifies what to put before the first column alignment specifier.

```
\newcommand*{\dtlbeforecols}{}
```

\dtlaftercols This specifies what to put after the last column alignment specifier.

```
\newcommand*{\dtlaftercols}{}
```

\dtlstringalign Alignment character for columns containing strings

```
\newcommand*{\dtlstringalign}{l}
```

\dtlintalign Alignment character for columns containing integers

```
\newcommand*{\dtlintalign}{r}
```

\dtlrealalign Alignment character for columns containing real numbers

```
\newcommand*{\dtlrealalign}{r}
```

tlcurrencyalign Alignment character for columns containing currency numbers

```
\newcommand*{\dtlcurrencyalign}{r}
```

\dtladdalign

```
\dtladdalign{⟨cs⟩}{⟨type⟩}{⟨col num⟩}{⟨max cols⟩}
```

Adds tabular column alignment character to ⟨*cs*⟩ for column ⟨*col num*⟩ which contains data type ⟨*type*⟩.

```
\newcommand*{\dtladdalign}[4]{%
  \ifnum#3=1\relax
    \protected@edef#1{\dtlbeforecols}%
  \else
    \protected@edef#1{#1\dtlbetweencols}%
  \fi
  \ifstrempty{#2}%
  {%
    \protected@edef#1{#1c}%
  }%
  {%
    \ifcase#2\relax
```

string

```
      \protected@edef#1{#1\dtlstringalign}%
    \or
```

integer

```
      \protected@edef#1{#1\dtlintalign}%
    \or
```

real number

```
      \protected@edef#1{#1\dtlrealalign}%
    \or
```

currency

```
      \protected@edef#1{#1\dtlcurrencyalign}%
    \else
```

Unknown type

```
      \protected@edef#1{#1c}%
      \PackageError{datatool}{Unknown data type ‘#2’}{}%
    \fi
  }%
  \ifnum#3=#4\relax
    \protected@edef#1{#1\dtlaftercols}%
  \fi
}
```

\dtlheaderformat

```
\dtlheaderformat{⟨text⟩}
```

Specifies how to format the column title.

```
\newcommand*{\dtlheaderformat}[1]{\null\hfil\textbf{#1}\hfil\null}
```

\dtlstringformat

\dtlstringformat{⟨*text*⟩}

Specifies how to format entries in columns with string data type.

```
\newcommand*{\dtlstringformat}[1]{#1}
```

\dtlintformat

\dtlintformat{⟨*text*⟩}

Specifies how to format entries in columns with integer data type.

```
\newcommand*{\dtlintformat}[1]{#1}
```

\dtlrealformat

\dtlrealformat{⟨*text*⟩}

Specifies how to format entries in columns with real data type.

```
\newcommand*{\dtlrealformat}[1]{#1}
```

tlcurrencyformat

\dtlcurrencyformat{⟨*text*⟩}

Specifies how to format entries in columns with currency data type.

```
\newcommand*{\dtlcurrencyformat}[1]{#1}
```

displaystarttab Indicates what to do just after \begin{tabular}{⟨*column specs*⟩} (e.g. \hline).

```
\newcommand*{\dtldisplaystarttab}{}
```

tldisplayendtab Indicates what to do just before \end{tabular}.

```
\newcommand*{\dtldisplayendtab}{}
```

isplayafterhead Indicates what to do after the header row, before the first row of data.

```
\newcommand*{\dtldisplayafterhead}{}
```

tldisplayvalign Stores the vertical alignment specifier for the tabular environment used in \DTLdisplaydb

```
\newcommand*{\dtldisplayvalign}{c}
```

displaystartrow Indicates what to do at the start of each row (not including the header row or the first row of data).

```
\newcommand*{\dtldisplaystartrow}{}
```

\dtldisplaycr

```
\newcommand{\dtldisplaycr}{\tabularnewline}
```

\DTLdisplaydb

```
\DTLdisplaydb[⟨omit list⟩]{⟨db⟩}
```

Displays the database ⟨*db*⟩ in a tabular environment.

```
\newcommand*{\DTLdisplaydb}[2][]{%
```

Initialise: only want & between columns

```
  \def\@dtl@doamp{\gdef\@dtl@doamp{&}}%
  \def\@dtl@resetdoamp{\gdef\@dtl@doamp{\gdef\@dtl@doamp{&}}}%
```

Store maximum number of columns

```
  \edef\@dtl@maxcols{\expandafter\number
    \csname dtlcols@#2\endcsname}%
```

Subtract number of omitted columns

```
  \DTLnumitemsinlist{#1}{\@dtl@tmp}%
  \dtlsub{\@dtl@maxcols}{\@dtl@maxcols}{\@dtl@tmp}%
  \dtlclip{\@dtl@maxcols}{\@dtl@maxcols}%
```

Argument for tabular environment

```
  \def\@dtl@tabargs{}%
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{#1}%
    {}%
    {%
       \dtladdalign\@dtl@tabargs\@dtl@type\@dtl@idx\@dtl@maxcols

    }%
  }%
```

Begin tabular environment

```
  \edef\@dtl@dobegintab{\noexpand\begin{tabular}[\dtldisplayvalign]{\@dtl@tabargs}}%
  \@dtl@dobegintab
```

Do start hook

```
  \dtldisplaystarttab
```

Reset `\@dtl@doamp` so it doesn't do an ampersand at the start of the first column.

```
  \@dtl@resetdoamp
```

Do the header row.

```
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{#1}%
    {}%
    {%
      \@dtl@doamp
      \dtlheaderformat{\@dtl@head}%
      }%
    }%
    \\%
```

Do the after header hook

```
    \dtldisplayafterhead
```

Reset \@dtl@doamp so it doesn't do an ampersand at the start of the first column.

```
    \@dtl@resetdoamp
```

Iterate through each row of the database

```
    \@sDTLforeach{#2}{}{%
```

Do the start row hook if not the first row

```
      \DTLiffirstrow{}{\dtldisplaycr\dtldisplaystartrow}%
```

Reset \@dtl@doamp so it doesn't do an ampersand at the start of the first column.

```
      \@dtl@resetdoamp
```

Iterate through each column.

```
      \DTLforeachkeyinrow{\@dtl@val}%
      {%
        \expandafter\DTLifinlist\expandafter{\dtlkey}{#1}%
        {}%
        {%
```

Need to make value global as it needs to be used after the ampersand.

```
          \global\let\@dtl@val\@dtl@val
          \@dtl@doamp
```

\DTLforeachkeyinrow sets \dtltype to the data type for the current key. This can be used to determine which format to use for this entry.

```
          \@dtl@datatype=0\dtltype\relax
          \ifcase\@dtl@datatype
            \dtlstringformat\@dtl@val
          \or
            \dtlintformat\@dtl@val
          \or
            \dtlrealformat\@dtl@val
          \or
            \dtlcurrencyformat\@dtl@val
          \else
            \@dtl@val
          \fi
        }%
      }%
    }%
    \dtldisplayendtab
    \end{tabular}%
  }
```

Define keys to use in the optional argument of \DTLdisplaylongdb.

The caption key sets the caption for the longtable.

```
\define@key{displaylong}{caption}{\def\@dtl@cap{#1}}
```

The contcaption key sets the continuation caption for the longtable.

```
\define@key{displaylong}{contcaption}{\def\@dtl@contcap{#1}}
```

The shortcaption key sets the lof caption for the longtable.

```
\define@key{displaylong}{shortcaption}{\def\@dtl@shortcap{#1}}
```

The label key sets the label for the longtable.

```
\define@key{displaylong}{label}{\def\@dtl@label{#1}}
```

The foot key sets the longtable foot

```
\define@key{displaylong}{foot}{\def\@dtl@foot{#1}}
```

The lastfoot key sets the longtable last foot

```
\define@key{displaylong}{lastfoot}{\def\@dtl@lastfoot{#1}}
```

List of omitted columns

```
\define@key{displaylong}{omit}{\def\@dtl@omitlist{#1}}
```

resetdostartrow Resets start row hook so that it skips the first row.

```
\newcommand*{\@dtl@resetdostartrow}{%
  \gdef\@dtl@dostartrow{%
    \gdef\@dtl@dostartrow{\dtldisplaycr\dtldisplaystartrow}}%
}
```

DTLdisplaylongdb

\DTLdisplaylongdb[⟨*options*⟩]{⟨*db*⟩}

Displays the database ⟨*db*⟩ in a longtable environment. (User needs to load longtable).

```
\newcommand*{\DTLdisplaylongdb}[2][]{%
```

Initialise.

```
  \def\@dtl@cap{\@nil}%
  \def\@dtl@contcap{\@nil}%
  \def\@dtl@label{\@nil}%
  \def\@dtl@shortcap{\@dtl@cap}%
  \def\@dtl@foot{\@nil}%
  \def\@dtl@lastfoot{\@nil}%
  \def\@dtl@omitlist{}%
```

Set the options

```
  \setkeys{displaylong}{#1}%
```

Only want & between columns

```
  \def\@dtl@doamp{\gdef\@dtl@doamp{&}}%
  \def\@dtl@resetdoamp{\gdef\@dtl@doamp{\gdef\@dtl@doamp{&}}}%
  \@dtl@resetdostartrow
```

Store maximum number of columns

```
\edef\@dtl@maxcols{\expandafter\number
  \csname dtlcols@#2\endcsname}%
```

Subtract number of omitted columns

```
\DTLnumitemsinlist{\@dtl@omitlist}{\@dtl@tmp}%
\dtlsub{\@dtl@maxcols}{\@dtl@maxcols}{\@dtl@tmp}%
\dtlclip{\@dtl@maxcols}{\@dtl@maxcols}%
```

Argument for longtable environment

```
\def\@dtl@tabargs{}%
\dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
  \in{#2}\do
{%
  \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
  {}%
  {%
    \dtladdalign\@dtl@tabargs\@dtl@type\@dtl@idx\@dtl@maxcols
  }%
}%
```

Start the longtable environment.

```
\edef\@dtl@dobegintab{\noexpand\begin{longtable}{\@dtl@tabargs}}%
\@dtl@dobegintab
```

Is a foot required?

```
\ifx\@dtl@foot\@nnil
\else
  \@dtl@foot\endfoot
\fi
```

Is a last foot required?

```
\ifx\@dtl@lastfoot\@nnil
\else
  \@dtl@lastfoot\endlastfoot
\fi
```

Is a caption required?

```
\ifx\@dtl@cap\@nnil
```

No caption required, just do header row.

```
  \@dtl@resetdoamp
  \dtldisplaystarttab
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
    {}%
    {%
      \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
    }%
  }%
  \@dtl@resetdoamp
  \@dtl@resetdostartrow
  \endhead\dtldisplayafterhead
\else
```

Caption is required

```
  \caption[\@dtl@shortcap]{\@dtl@cap}%
```

Is a label required?

```
  \ifx\@dtl@label\@nnil
  \else
    \label{\@dtl@label}%
  \fi
  \dtldisplaycr
```

Do start hook.

```
\dtldisplaystarttab
```

Do header row.

```
  \@dtl@resetdoamp
  \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
  {%
    \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
    {}%
    {%
      \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
    }%
  }%
  \@dtl@resetdoamp
```

```
  \dtldisplaycr\dtldisplayafterhead
  \endfirsthead
```

Is a continuation caption required?

```
  \ifx\@dtl@contcap\@nnil
    \caption{\@dtl@cap}%
  \else
    \caption{\@dtl@contcap}%
  \fi
```

Do start hook.

```
  \dtldisplaycr\dtldisplaystarttab
```

Do header row.

```
    \@dtl@resetdoamp
    \dtlforeachkey(\@dtl@key,\@dtl@idx,\@dtl@type,\@dtl@head)%
    \in{#2}\do
    {%
      \expandafter\DTLifinlist\expandafter{\@dtl@key}{\@dtl@omitlist}%
      {}%
      {%
```

```
          \@dtl@doamp{\dtlheaderformat{\@dtl@head}}%
        }%
      }%
      \@dtl@resetdoamp
      \@dtl@resetdostartrow

    \dtldisplaycr\dtldisplayafterhead
    \endhead
  \fi
```

Iterate through each row of the database

```
  \@sDTLforeach{#2}{}{%
    \@dtl@dostartrow
    \@dtl@resetdoamp
```

Iterate through each column

```
    \DTLforeachkeyinrow{\@dtl@val}%
    {%
      \global\let\@dtl@val\@dtl@val
      \expandafter\DTLifinlist\expandafter{\dtlkey}{\@dtl@omitlist}%
      {}%
      {%
        \@dtl@doamp
```

`\DTLforeachkeyinrow` sets `\dtltype` to the data type for the current key. This can be used to determine which format to use for this entry.

```
        \@dtl@datatype=0\dtltype\relax
        \ifcase\@dtl@datatype
          \dtlstringformat\@dtl@val
        \or
          \dtlintformat\@dtl@val
        \or
          \dtlrealformat\@dtl@val
        \or
          \dtlcurrencyformat\@dtl@val
        \fi
      }%
    }%
  }%
  \dtldisplayendtab
  \end{longtable}%
}
```

4.8 Editing Databases

\dtlswaprows

`\dtlswaprows{⟨db⟩}{⟨row1 idx⟩}{⟨row2 idx⟩}`

Swaps the rows with indices ⟨*row1 idx*⟩ and ⟨*row2 idx*⟩ in the database ⟨*db*⟩. (Doesn't check if data base exists or if indices are out of bounds.)

```
\newcommand*{\dtlswaprows}[3]{%
  \ifnum#2=#3\relax
```

Attempt to swap row with itself: do nothing.

```
  \else
```

Let row A be the row with the lower index and row B be the row with ther higher index.

```
    \ifnum#2<#3\relax
      \edef\@dtl@rowAidx{\number#2}%
      \edef\@dtl@rowBidx{\number#3}%
    \else
      \edef\@dtl@rowAidx{\number#3}%
      \edef\@dtl@rowBidx{\number#2}%
    \fi
```

Split the database around row A.

```
    \edef\@dtl@dosplit{\noexpand\dtlgetrow{#1}{\@dtl@rowAidx}}%
    \@dtl@dosplit
```

Store first part of database in `\@dtl@firstpart`.

```
    \expandafter\def\expandafter\@dtl@firstpart\expandafter
      {\the\dtlbeforerow}%
```

Store row A in `\@dtl@toksA`.

```
    \@dtl@toksA=\dtlcurrentrow
```

Split the second part (everything after row A).

```
    \edef\@dtl@dosplit{\noexpand\@dtlgetrow
      {\the\dtlafterrow}{\@dtl@rowBidx}}%
    \@dtl@dosplit
```

Store the mid part (everything between row A and row B)

```
    \expandafter\def\expandafter\@dtl@secondpart\expandafter
      {\the\dtlbeforerow}%
```

Store row B in `\@dtl@toksB`.

```
    \@dtl@toksB=\dtlcurrentrow
```

Store the last part (everything after row B).

```
    \expandafter\def\expandafter\@dtl@thirdpart\expandafter
      {\the\dtlafterrow}%
```

Reconstruct database: store first part in `\toks@`

```
    \toks@=\expandafter{\@dtl@firstpart}%
```

Store mid part in `\dtl@toks`

```
    \@dtl@toks=\expandafter{\@dtl@secondpart}%
```

Format data for first part, row B and mid part.

```
    \edef\@dtl@tmp{\the\toks@
     \noexpand\db@row@elt@w%
     \noexpand\db@row@id@w \@dtl@rowAidx\noexpand\db@row@id@end@%
     \the\@dtl@toksB
     \noexpand\db@row@id@w \@dtl@rowAidx\noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
     \the\@dtl@toks}%
```

Store data so far in `\toks@`.

```
    \toks@=\expandafter{\@dtl@tmp}%
```

Store last part in `\dtl@toks`.

```
    \@dtl@toks=\expandafter{\@dtl@thirdpart}%
```

Format row A and end part.

```
    \edef\@dtl@tmp{\the\toks@
     \noexpand\db@row@elt@w%
     \noexpand\db@row@id@w \@dtl@rowBidx\noexpand\db@row@id@end@%
     \the\@dtl@toksA
     \noexpand\db@row@id@w \@dtl@rowBidx\noexpand\db@row@id@end@%
     \noexpand\db@row@elt@end@%
    \the\@dtl@toks}%
```

Update the database

```
    \expandafter\global\csname dtldb@#1\endcsname=\expandafter
      {\@dtl@tmp}%
  \fi
}
```

tl@decrementrows

`\dtl@decrementrows{⟨toks⟩}{⟨n⟩}`

decrement by 1 all rows in ⟨*toks*⟩ with row index above ⟨*n*⟩

```
\newcommand*{\dtl@decrementrows}[2]{%
  \def\@dtl@newlist{}%
  \edef\@dtl@min{\number#2}%
  \expandafter\@dtl@decrementrows\the#1%
    \db@row@elt@w%
      \db@row@id@w \@nil\db@row@id@end@%
      \db@row@id@w \@nil\db@row@id@end@%
    \db@row@elt@end@%
    \@nil
  #1=\expandafter{\@dtl@newlist}%
}
```

l@decrementrows

```
\def\@dtl@decrementrows\db@row@elt@w\db@row@id@w #1\db@row@id@end@%
#2\db@row@id@w #3\db@row@id@end@\db@row@elt@end@#4\@nil{%
  \def\@dtl@thisrow{#1}%
  \ifx\@dtl@thisrow\@nnil
    \let\@dtl@donextdec=\@dtl@gobbletonil
  \else
```

```
    \ifnum\@dtl@thisrow>\@dtl@min
      \@dtl@tmpcount=\@dtl@thisrow\relax
      \advance\@dtl@tmpcount by -1\relax
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \noexpand\db@row@elt@w% row header
        \noexpand\db@row@id@w \number\@dtl@tmpcount
          \noexpand\db@row@id@end@% row id
         \the\toks@ % row contents
        \noexpand\db@row@id@w \number\@dtl@tmpcount
          \noexpand\db@row@id@end@% row id
        \noexpand\db@row@elt@end@% row end
      }%
    \else
      \toks@{#2}%
      \@dtl@toks=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\@dtl@toks
        \noexpand\db@row@elt@w% row header
        \noexpand\db@row@id@w #1%
          \noexpand\db@row@id@end@% row id
         \the\toks@ % row contents
        \noexpand\db@row@id@w #3%
          \noexpand\db@row@id@end@% row id
        \noexpand\db@row@elt@end@% row end
      }%
    \fi
    \let\@dtl@donextdec=\@dtl@decrementrows
  \fi
  \@dtl@donextdec#4\@nil
}
```

\DTLremoverow

`\DTLremoverow{⟨db⟩}{⟨row index⟩}`

Remove row with given index from database named ⟨*db*⟩.

```
\newcommand*{\DTLremoverow}[2]{%
```

Check database exists

```
  \DTLifdbexists{#1}%
  {%
```

Check index if index is out of bounds

```
    \ifnum#2>0\relax
```

Check if data base has at least ⟨*row index*⟩ rows

```
      \expandafter\ifnum\csname dtlrows@#1\endcsname<#2\relax
        \expandafter\ifnum\csname dtlrows@#1\endcsname=1\relax
          \PackageError{datatool}{Can't remove row `\number#2' from
      database ‘#1’: no such row}{Database ‘#1’ only has
      1 row}%
    \else
     \PackageError{datatool}{Can’t remove row ‘\number#2’ from
      database ‘#1’: no such row}{Database ‘#1’ only has
      \expandafter\number\csname dtlrows@#1\endcsname\space
      rows}%
    \fi
   \else
    \@DTLremoverow{#1}{#2}%
   \fi
  \else
   \PackageError{datatool}{Can’t remove row \number#2: index
    out of bounds}{Row indices start at 1}%
  \fi
 }%
 {%
  \PackageError{datatool}{Can’t remove row: database ‘#1’ doesn’t
   exist}{}%
 }%
}
```

\@DTLremoverow

\@DTLremoverow{⟨*db*⟩}{⟨*row index*⟩}

Doesn’t perform any checks for the existence of the database or if the index is in range.

```
\newcommand*{\@DTLremoverow}[2]{%
```

Get row from data base

```
   \edef\dtl@dogetrow{\noexpand\dtlgetrow{#1}{\number#2}}%
   \dtl@dogetrow
```

Update the row indices

```
   \expandafter\dtl@decrementrows\expandafter
    {\dtlbeforerow}{#2}%
   \expandafter\dtl@decrementrows\expandafter
    {\dtlafterrow}{#2}%
```

Reconstruct database

```
   \edef\dtl@tmp{\the\dtlbeforerow \the\dtlafterrow}%
   \expandafter\global\csname dtldb@#1\endcsname
    =\expandafter{\dtl@tmp}%
```

decrement row counter

```
   \expandafter\global\expandafter\advance
    \csname dtlrows@#1\endcsname by -1\relax
}
```

4.9 Database Functions

\DTLsumforkeys

```
\DTLsumforkeys[⟨condition⟩][⟨assign list⟩]{⟨db list⟩}{⟨key list⟩}{⟨cmd⟩}
```

Sums all entries for key ⟨*key*⟩ over all databases listed in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument provides an assignment list to pass to \DTLforeach in case extra information is need by ⟨*condition*⟩.

```
\newcommand*{\DTLsumforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlsumforkeys
}
```

\@dtlsumforkeys

```
\newcommand*{\@dtlsumforkeys}[4][]{%
  \def#4{0}%
```

Iterate over all the listed data bases

```
  \@for\@dtl@db@name:=#2\do{%
```

Iterate through this database (using read only version)

```
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
```

Iterate through key list.

```
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
          {\DTLadd{#4}{#4}{\DTLthisval}}{}%
      }%
    }%
  }%
}
```

\DTLsumcolumn

```
\DTLsumcolumn{⟨db⟩}{⟨key⟩}{⟨cmd⟩}
```

Quicker version of \DTLsumforkeys that just sums over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
\newcommand*{\DTLsumcolumn}[3]{%
  \def#3{0}%
```

Check data base exists

```
  \DTLifdbexists{#1}%
  {%
```

Check column exists

```
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \DTLadd{#3}{#3}{\DTLthisval}%
       }%
     }%
```

key not defined for this data base

```
     {%
       \PackageError{datatool}{Key ‘#2’ doesn’t
         exist in database ‘#1’}{}%
     }%
  }%
```

data base doesn’t exist

```
  {%
    \PackageError{datatool}{Data base ‘#1’ doesn’t
      exist}{}%
  }%
}
```

\DTLmeanforkeys

\DTLmeanforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Computes the arithmetic mean of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument allows an assignment list to be passed to \DTLforeach.

```
\newcommand*{\DTLmeanforkeys}[1][\boolean{true}\and
 \DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlmeanforkeys
}
```

\@dtl@elements

Count register to keep track of number of elements

```
\newcount\@dtl@elements
```

@dtlmeanforkeys

```
\newcommand*{\@dtlmeanforkeys}[4][]{%
  \def#4{0}%
  \@dtl@elements=0\relax
```

Iterate over all the listed data bases

```
  \@for\@dtl@db@name:=#2\do{%
```

Iterate through this database (using read only version)

```
    \@sDTLforeach{\@dtl@db@name}%
    {#1}% assignment list
    {%
```

Iterate through key list.

```
      \@for\@dtl@key:=#3\do{%
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
        \dtlcurrentrow=\expandafter{\dtl@thisrow}%
        \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
        \expandafter\ifthenelse\expandafter{\@dtl@cond}%
        {%
          \DTLadd{#4}{#4}{\DTLthisval}%
          \advance\@dtl@elements by 1\relax
        }{}%
      }%
    }%
  }%
```

Divide total by number of elements summed.

```
  \ifnum\@dtl@elements=0\relax
    \PackageError{datatool}{Unable to evaluate mean: no data}{}%
  \else
    \edef\@dtl@n{\number\@dtl@elements}%
    \DTLdiv{#4}{#4}{\@dtl@n}%
  \fi
}
```

DTLmeanforcolumn

\DTLmeanforcolumn{⟨*db*⟩}{⟨*key*⟩}{⟨*cmd*⟩}

Quicker version of \DTLmeanforkeys that just computes the mean over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
\newcommand*{\DTLmeanforcolumn}[3]{%
  \def#3{0}%
  \@dtl@elements=0\relax
```

Check data base exists

```
  \DTLifdbexists{#1}%
  {%
```

Check column exists

```
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
```

```
           \DTLadd{#3}{#3}{\DTLthisval}%
           \advance\@dtl@elements by 1\relax
         }%
         \ifnum\@dtl@elements=0\relax
           \PackageError{datatool}{Can't compute mean for
            column '#2' in database '#1': no data}{}%
         \else
           \edef\@dtl@n{\number\@dtl@elements}%
           \DTLdiv{#3}{#3}{\@dtl@n}%
         \fi
       }%
```

key not defined for this data base

```
       {%
         \PackageError{datatool}{Key '#2' doesn't
           exist in database '#1'}{}%
       }%
    }%
```

data base doesn't exist

```
    {%
      \PackageError{datatool}{Data base '#1' doesn't
        exist}{}%
    }%
  }
```

Lvarianceforkeys

\DTLvarianceforkeys[⟨*condition*⟩][⟨*assign list*⟩]{⟨*db list*⟩}{⟨*key list*⟩}{⟨*cmd*⟩}

Computes the variance of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument is an assignment list to pass to \DTLforeach in case it is required for the condition.

```
  \newcommand*{\DTLvarianceforkeys}[1][\boolean{true}\and
   \DTLisnumerical{\DTLthisval}]{%
    \def\@dtl@cond{#1}%
    \@dtlvarianceforkeys
  }
```

@dtlmeanforkeys

```
  \newcommand*{\@dtlvarianceforkeys}[4][]{%
    \@dtlmeanforkeys[#1]{#2}{#3}{\dtl@mean}%
    \def#4{0}%
    \@dtl@elements=0\relax
```

Iterate over all the listed data bases

```
    \@for\@dtl@db@name:=#2\do{%
```

Iterate through this database (using read only version)

```
   \@sDTLforeach{\@dtl@db@name}%
   {#1}% assignment list
   {%
```

Iterate through key list.

```
    \@for\@dtl@key:=#3\do{%
     \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
     \dtlcurrentrow=\expandafter{\dtl@thisrow}%
     \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
     \expandafter\ifthenelse\expandafter{\@dtl@cond}%
     {%
```

compute $(x_i - \mu)^2$

```
       \DTLsub{\dtl@diff}{\DTLthisval}{\dtl@mean}%
       \DTLmul{\dtl@diff}{\dtl@diff}{\dtl@diff}%
       \DTLadd{#4}{#4}{\dtl@diff}%
       \advance\@dtl@elements by 1\relax
     }{}%
    }%
   }%
  }%
```

Divide by number of elements.

```
  \ifnum\@dtl@elements=0\relax
   \PackageError{datatool}{Unable to evaluate variance: no data}{}%
  \else
   \edef\@dtl@n{\number\@dtl@elements}%
   \DTLdiv{#4}{#4}{\@dtl@n}%
  \fi
 }
```

arianceforcolumn

\DTLvarianceforcolumn{⟨*db*⟩}{⟨*key*⟩}{⟨*cmd*⟩}

Quicker version of \DTLvarianceforkeys that just computes the variance over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
 \newcommand*{\DTLvarianceforcolumn}[3]{%
  \DTLmeanforcolumn{#1}{#2}{\dtl@mean}%
  \def#3{0}%
  \@dtl@elements=0\relax
```

Check data base exists

```
  \DTLifdbexists{#1}%
  {%
```

Check column exists

```
   \@sDTLifhaskey{#1}{#2}%
   {%
```

```
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
```

compute $(x_i - \mu)^2$

```
         \DTLsub{\dtl@diff}{\DTLthisval}{\dtl@mean}%
         \DTLmul{\dtl@diff}{\dtl@diff}{\dtl@diff}%
         \DTLadd{#3}{#3}{\dtl@diff}%
         \advance\@dtl@elements by 1\relax
       }%
       \ifnum\@dtl@elements=0\relax
         \PackageError{datatool}{Can't compute variance for
          column `#2' in database `#1': no data}{}%
       \else
         \edef\@dtl@n{\number\@dtl@elements}%
         \DTLdiv{#3}{#3}{\@dtl@n}%
       \fi
     }%
```

key not defined for this data base

```
     {%
       \PackageError{datatool}{Key `#2' doesn't
         exist in database `#1'}{}%
     }%
  }%
```

data base doesn't exist

```
  {%
    \PackageError{datatool}{Data base `#1' doesn't
      exist}{}%
  }%
}
```

\DTLsdforkeys

`\DTLsdforkeys[⟨condition⟩][⟨assign list⟩]{⟨db list⟩}{⟨key list⟩}{⟨cmd⟩}`

Computes the standard deviation of all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for `\DTLforeach`. The second optional argument is an assignment list for `\DTLforeach` in case it is needed for the condition.

```
\newcommand*{\DTLsdforkeys}[1][\boolean{true}\and
\DTLisnumerical{\DTLthisval}]{%
  \def\@dtl@cond{#1}%
  \@dtlsdforkeys
}
```

\@dtlsdforkeys

```
\newcommand*{\@dtlsdforkeys}[4][]{%
  \@dtlvarianceforkeys[#1]{#2}{#3}{#4}%
```

```
    \DTLsqrt{#4}{#4}%
  }
```

\DTLsdforcolumn

```
\DTLsdforcolumn{⟨db⟩}{⟨key⟩}{⟨cmd⟩}
```

Quicker version of \DTLsdforkeys that just computes the standard deviation over one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
  \newcommand*{\DTLsdforcolumn}[3]{%
    \DTLvarianceforcolumn{#1}{#2}{#3}%
    \DTLsqrt{#3}{#3}%
  }
```

\DTLminforkeys

```
\DTLminforkeys[⟨condition⟩][⟨assign list⟩]{⟨db list⟩}{⟨key list⟩}{⟨cmd⟩}
```

Determines the minimum over all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument is an assignment list for \DTLforeach in the event that extra information is need for the condition.

```
  \newcommand*{\DTLminforkeys}[1][\boolean{true}\and
   \DTLisnumerical{\DTLthisval}]{%
    \def\@dtl@cond{#1}%
    \@dtlminforkeys
  }
```

\@dtlminforkeys

```
  \newcommand*{\@dtlminforkeys}[4][]{%
    \def#4{}%
```

Iterate over all the listed data bases

```
    \@for\@dtl@db@name:=#2\do{%
```

Iterate through this database (using read only version)

```
      \@sDTLforeach{\@dtl@db@name}%
      {#1}% assignment list
      {%
```

Iterate through key list.

```
        \@for\@dtl@key:=#3\do{%
          \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
          \dtlcurrentrow=\expandafter{\dtl@thisrow}%
          \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
          \expandafter\ifthenelse\expandafter{\@dtl@cond}%
          {%
            \ifdefempty{#4}%
            {%
              \let#4\DTLthisval
            }%
            {%
              \DTLmin{#4}{#4}{\DTLthisval}%
            }%
          }{}%
        }%
      }%
    }%
  }
```

\DTLminforcolumn

`\DTLminforcolumn{⟨db⟩}{⟨key⟩}{⟨cmd⟩}`

Quicker version of `\DTLminforkeys` that just finds the minimum value in one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
\newcommand*{\DTLminforcolumn}[3]{%
  \def#3{}%
```

Check data base exists

```
  \DTLifdbexists{#1}%
  {%
```

Check column exists

```
     \@sDTLifhaskey{#1}{#2}%
     {%
       \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
       {%
         \ifdefempty{#3}%
         {%
           \let#3\DTLthisval
         }%
         {%
           \DTLmin{#3}{#3}{\DTLthisval}%
         }%
       }%
     }%
```

key not defined for this data base

```
     {%
       \PackageError{datatool}{Key ‘#2’ doesn’t
         exist in database ‘#1’}{}%
     }%
  }%
```

data base doesn’t exist

```
  {%
```

```
      \PackageError{datatool}{Data base ‘#1’ doesn’t
        exist}{}%
    }%
  }
```

\DTLmaxforkeys

```
\DTLmaxforkeys[⟨condition⟩][⟨assign list⟩]{⟨db list⟩}{⟨key list⟩}{⟨cmd⟩}
```

Determines the maximum over all entries for each key in ⟨*key list*⟩ over all databases in ⟨*db list*⟩, and stores in ⟨*cmd*⟩, which must be a control sequence. The first optional argument ⟨*condition*⟩ is the same as that for \DTLforeach. The second optional argument is an assignment list to pass to \DTLforeach in the event that extra information is required in the condition.

```
  \newcommand*{\DTLmaxforkeys}[1][\boolean{true}\and
   \DTLisnumerical{\DTLthisval}]{%
    \def\@dtl@cond{#1}%
    \@dtlmaxforkeys
  }
```

\@dtlmaxforkeys

```
  \newcommand*{\@dtlmaxforkeys}[4][]{%
    \def#4{}%
```

Iterate over all the listed data bases

```
    \@for\@dtl@db@name:=#2\do{%
```

Iterate through this database (using read only version)

```
      \@sDTLforeach{\@dtl@db@name}%
      {#1}% assignment list
      {%
```

Iterate through key list.

```
        \@for\@dtl@key:=#3\do{%
          \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@db@name}{\@dtl@key}%
          \dtlcurrentrow=\expandafter{\dtl@thisrow}%
          \dtlgetentryfromrow{\DTLthisval}{\@dtl@col}{\dtlcurrentrow}%
          \expandafter\ifthenelse\expandafter{\@dtl@cond}%
          {%
            \ifdefempty{#4}%
            {%
              \let#4\DTLthisval
            }%
            {%
              \DTLmax{#4}{#4}{\DTLthisval}%
            }%
          }{}%
        }%
      }%
```

```
    }%
  }
```

\DTLmaxforcolumn

```
\DTLmaxforcolumn{⟨db⟩}{⟨key⟩}{⟨cmd⟩}
```

Quicker version of \DTLmaxforkeys that just finds the maximum value in one column (specified by ⟨*key*⟩) for a single database (specified by ⟨*db*⟩) and stores the result in ⟨*cmd*⟩.

```
  \newcommand*{\DTLmaxforcolumn}[3]{%
    \def#3{}%
```

Check data base exists

```
    \DTLifdbexists{#1}%
    {%
```

Check column exists

```
      \@sDTLifhaskey{#1}{#2}%
      {%
        \@sdtlforcolumn{\DTLthisval}{#1}{#2}%
        {%
          \ifdefempty{#3}%
          {%
            \let#3\DTLthisval
          }%
          {%
            \DTLmax{#3}{#3}{\DTLthisval}%
          }%
        }%
      }%
```

key not defined for this data base

```
      {%
        \PackageError{datatool}{Key ‘#2’ doesn’t
          exist in database ‘#1’}{}%
      }%
    }%
```

data base doesn’t exist

```
    {%
      \PackageError{datatool}{Data base ‘#1’ doesn’t
        exist}{}%
    }%
  }
```

DTLcomputebounds

```
\DTLcomputebounds[⟨condition⟩]{⟨db list⟩}{⟨x key⟩}{⟨y key⟩}{⟨minX cmd⟩}
{⟨minY cmd⟩}{⟨maxX cmd⟩}{⟨maxY cmd⟩}
```

Computes the maximum and minimum *x* and *y* values over all the databases listed in ⟨*db list*⟩ where the *x* value is given by ⟨*x key*⟩ and the *y* value is given by ⟨*y key*⟩. The results are stored in ⟨*minX cmd*⟩, ⟨*minY cmd*⟩, ⟨*maxX cmd*⟩ and ⟨*maxY cmd*⟩ in standard decimal format.

```
\newcommand*{\DTLcomputebounds}[8][\boolean{true}]{%
\let#5=\relax
\let#6=\relax
\let#7=\relax
\let#8=\relax
\@for\dtl@thisdb:=#2\do{%
  \@sDTLforeach[#1]{\dtl@thisdb}{\DTLthisX=#3,\DTLthisY=#4}{%
    \expandafter\DTLconverttodecimal\expandafter{\DTLthisX}{\dtl@decx}%
    \expandafter\DTLconverttodecimal\expandafter{\DTLthisY}{\dtl@decy}%
    \ifx#5\relax
      \let#5=\dtl@decx
      \let#6=\dtl@decy
      \let#7=\dtl@decx
      \let#8=\dtl@decy
    \else
      \dtlmin{#5}{#5}{\dtl@decx}%
      \dtlmin{#6}{#6}{\dtl@decy}%
      \dtlmax{#7}{#7}{\dtl@decx}%
      \dtlmax{#8}{#8}{\dtl@decy}%
    \fi
  }%
}%
}
```

TLgetvalueforkey

`\DTLgetvalueforkey{⟨cmd⟩}{⟨key⟩}{⟨db name⟩}{⟨ref key⟩}{⟨ref value⟩}`

This (globally) sets ⟨*cmd*⟩ (a control sequence) to the value of the key specified by ⟨*key*⟩ in the first row of the database called ⟨*db name*⟩ which contains the key ⟨*ref key*⟩ which has the value ⟨*value*⟩.

```
\newcommand*{\DTLgetvalueforkey}[5]{%
```

Get row containing referenced (key,value) pair

```
  \DTLgetrowforkey{\@dtl@row}{#3}{#4}{#5}%
```

Get column number for ⟨*key*⟩

```
  \@sdtl@getcolumnindex{\@dtl@col}{#3}{#2}%
```

Get value for given column

```
  {%
    \dtlcurrentrow=\expandafter{\@dtl@row}%
    \edef\@dtl@dogetval{\noexpand\dtlgetentryfromcurrentrow
      {\noexpand\@dtl@val}{\@dtl@col}}%
    \@dtl@dogetval
    \global\let#1=\@dtl@val
  }%
}
```

\DTLgetrowforkey

```
\DTLgetrowforkey{⟨cmd⟩}{⟨db name⟩}{⟨ref key⟩}{⟨ref value⟩}
```

This (globally) sets ⟨*cmd*⟩ (a control sequence) to the first row of the database called ⟨*db name*⟩ which contains the key ⟨*ref key*⟩ that has the value ⟨*value*⟩.

```
\newcommand*{\DTLgetrowforkey}[4]{%
  \global\let#1=\@empty
  \@sDTLforeach{#2}{\dtl@refvalue=#3}{%
    \DTLifnull{\dtl@refvalue}%
    {}%
    {%
      \ifthenelse{\equal{\dtl@refvalue}{#4}}%
      {%
        \xdef#1{\the\dtlcurrentrow}%
        \dtlbreak
      }%
      {}%
    }%
  }%
}
```

4.10 Sorting Databases

\@dtl@list Token register to store data when sorting.

```
\newtoks\@dtl@list
```

\DTLsort

```
\DTLsort[⟨replacement keys⟩]{⟨sort criteria⟩}{⟨db name⟩}
```

Sorts database ⟨*db name*⟩ according to {⟨*sort criteria*⟩}, which must be a comma separated list of keys, and optionally =⟨*order*⟩, where ⟨*order*⟩ is either `ascending` or `descending`. The optional argument is a list of keys to uses if the given key has a null value. The starred version uses a case insensitive string comparison.

```
\newcommand*{\DTLsort}{\@ifstar\@sDTLsort\@DTLsort}
```

\@DTLsort Unstarred (case sensitive) version.

```
\newcommand{\@DTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlcompare}%
}
```

\@sDTLsort Starred (case insensitive) version.

```
\newcommand*{\@sDTLsort}[3][]{%
  \dtlsort[#1]{#2}{#3}{\dtlicompare}%
}
```

\dtlsort

\dtlsort[⟨*replacement keys*⟩]{⟨*sort criteria*⟩}{⟨*db name*⟩}{⟨*handler*⟩}

More general version where user supplies a handler for the comparison.

```
\newcommand{\dtlsort}[4][]{%
```

Check the database exists

```
  \DTLifdbexists{#3}%
  {%
     \ifnum\DTLrowcount{#3}>100\relax
       \typeout{Sorting `#3' - this may take a while.}%
     \fi
```

Store replacement keys in \@dtl@replacementkeys.

```
    \edef\@dtl@replacementkeys{#1}%
```

Store sort order in \@dtl@sortorder, but check specified keys exist.

```
    \def\@dtl@sortorder{}%
    \@for\@dtl@level:=#2\do
    {%
```

Get key (stored in \@dtl@key).

```
      \expandafter\@dtl@getsortdirection\@dtl@level=\relax
```

Check key exists.

```
      \DTLifhaskey{#3}{\@dtl@key}%
      {%
```

Key exists, so add to \@dtl@sortorder.

```
         \ifdefempty\@dtl@sortorder
         {\let\@dtl@sortorder=\@dtl@level}%
         {\eappto\@dtl@sortorder{,\@dtl@level}}%
      }%
      {%
```

Key doesn't exist.

```
         \PackageError{datatool}%
         {%
            Can't sort on `\@dtl@level'.
            No such key `\@dtl@key' in database `#3'%
         }{}%
      }%
    }%
```

Now check if we have any keys left to sort on.

```
      \ifdefempty\@dtl@sortorder
      {%
         \PackageWarning{datatool}{No keys provided to sort database ‘#3’}%
      }%
      {%
```

Set `\@dtl@comparecs` to the required string comparison function. (Using case insensitive comparison macro `\dtlicompare`.)

```
        \let\@dtl@comparecs=#4%
```

Sort the database.

```
        \dtl@sortdata{#3}%
      }%
    }%
    {%
      \PackageError{datatool}{Database ‘#3’ doesn’t exist}{}%
    }%
  }
```

`\@dtl@rowa` Token register to store first row when sorting.

```
  \newtoks\@dtl@rowa
```

`\@dtl@rowb` Token register to store comparison row when sorting.

```
  \newtoks\@dtl@rowb
```

`\dtl@sortdata` `\dtl@sortdata{⟨db⟩}`

Sorts the data in named database using an insertion sort algorithm. `\@dtl@replacementkeys`, `\@dtl@sortorder` and `\@dtl@comparecs` must be set prior to use.

```
  \newcommand*{\dtl@sortdata}[1]{%
```

Initialise macro containing sorted data.

```
    \def\@dtl@sortedlist{}%
```

Store database name.

```
    \edef\@dtl@dbname{#1}%
```

Iterate through each row and insert into sorted list.

```
    \@dtlforeachrow(\@dtl@rowAnum,\@dtl@rowAcontents)\in\@dtl@dbname\do{%
      \@dtl@rowa=\expandafter{\@dtl@rowAcontents}%
```

Create a temporary list

```
      \def\@dtl@newlist{}%
```

Initialise the insertion for this iteration. Insertion hasn’t been done yet.

```
    \@dtl@insertdonefalse
```

Initialise row index to 0

```
    \dtlrownum=0\relax
```

Iterate through sorted list.

```
    \expandafter\@dtl@foreachrow\@dtl@sortedlist
      \db@row@elt@w%
      \db@row@id@w \@nil\db@row@id@end@%
      \db@row@id@w \@nil\db@row@id@end@%
      \db@row@elt@end@%
      \@@{\@dtl@rowBnum}{\@dtl@rowBcontents}%
      {%
```

Store row B in a token register

```
        \@dtl@rowb=\expandafter{\@dtl@rowBcontents}%
```

Get current row number of sorted list

```
        \dtlrownum=\@dtl@rowBnum
```

Has the insertion been done?

```
        \if@dtl@insertdone
```

New element has already been inserted, so just increment the row number to compensate for the inserted row.

```
          \advance\dtlrownum by 1\relax
        \else
```

Insertion hasn't been done yet. Compare row A and row B.

```
          \@dtl@sortcriteria{\@dtl@rowa}{\@dtl@rowb}%
```

If \dtl@sortresult is negative insert A before B.

```
          \ifnum\dtl@sortresult<0\relax
```

Insert row A into new list. First store \@dtl@newlist in \toks@.

```
            \toks@=\expandafter{\@dtl@newlist}%
```

Update \@dtl@newlist to be the old value followed by row A.

```
            \edef\@dtl@newlist{%
```

Old value:

```
              \the\toks@
```

Format row A

```
              \noexpand\db@row@elt@w%
               \noexpand\db@row@id@w \number\dtlrownum
               \noexpand\db@row@id@end@%
               \the\@dtl@rowa
               \noexpand\db@row@id@w \number\dtlrownum
               \noexpand\db@row@id@end@%
              \noexpand\db@row@elt@end@%
            }%
```

Increment row number to compensate for inserted row.

```
            \advance\dtlrownum by 1\relax
```

Mark insertion done.

```
          \@dtl@insertdonetrue
        \fi
      \fi
```

Insert row B

```
      \toks@=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\toks@
```

row B

```
        \noexpand\db@row@elt@w%
         \noexpand\db@row@id@w \number\dtlrownum
         \noexpand\db@row@id@end@%
         \the\@dtl@rowb
         \noexpand\db@row@id@w \number\dtlrownum
         \noexpand\db@row@id@end@%
        \noexpand\db@row@elt@end@%
      }%
```

Repeat loop.

```
    }\q@nil
```

If row A hasn't been inserted, do so now.

```
    \if@dtl@insertdone
    \else
```

`\dtlrownum` contains the index of the last row in new list, So increment it to get the new index for row A.

```
      \advance\dtlrownum by 1\relax
```

Insert row A.

```
      \toks@=\expandafter{\@dtl@newlist}%
      \edef\@dtl@newlist{\the\toks@
```

row A

```
        \noexpand\db@row@elt@w%
         \noexpand\db@row@id@w \number\dtlrownum
         \noexpand\db@row@id@end@%
         \the\@dtl@rowa
         \noexpand\db@row@id@w \number\dtlrownum
         \noexpand\db@row@id@end@%
        \noexpand\db@row@elt@end@%
      }%
    \fi
```

Set sorted list to new list.

```
    \let\@dtl@sortedlist=\@dtl@newlist
  }%
```

Update database.

```
  \expandafter\global\csname dtldb@#1\endcsname=\expandafter
    {\@dtl@sortedlist}%
}
```

dtl@sortcriteria

`\@dtl@sortcriteria{⟨row a toks⟩}{⟨row b toks⟩}`

`\@dtl@dbname` and `\@dtl@sortorder` must be set before use `\@dtl@sortorder` is a comma separated list of either just keys or ⟨*key*⟩=⟨*direction*⟩. (Check keys are valid before use.)

```
\newcommand{\@dtl@sortcriteria}[2]{%
```

Iterate through the sort order.

```
  \@for\@dtl@level:=\@dtl@sortorder\do
  {%
```

Set `\@dtl@sortdirection` to −1 (ascending) or +1 (descending). Key is stored in `\@dtl@key`.

```
    \expandafter\@dtl@getsortdirection\@dtl@level=\relax
```

Initially comparing on the same key

```
    \let\@dtl@keya=\@dtl@key
    \let\@dtl@keyb=\@dtl@key
```

Get values corresponding to key from both rows. First get column index corresponding to key.

```
    \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@key}%
```

Get entry for this column from row A and store in `\@dtl@a`.

```
    \dtlgetentryfromrow{\@dtl@a}{\@dtl@col}{#1}%
```

Get entry for this column from row B and store in `\@dtl@b`.

```
    \dtlgetentryfromrow{\@dtl@b}{\@dtl@col}{#2}%
```

Has value from row A been defined?

```
    \ifx\@dtl@a\dtlnovalue
```

Value hasn't been defined so set to null

```
      \@dtl@setnull{\@dtl@a}{\@dtl@key}%
    \fi
```

Has value from row B been defined?

```
    \ifx\@dtl@b\dtlnovalue
```

Value hasn't been defined so set to null

```
      \@dtl@setnull{\@dtl@b}{\@dtl@key}%
    \fi
```

Check if value for row A is null.

```
    \DTLifnull{\@dtl@a}%
    {%
```

Value for row A is null, so find the first non null key in list of replacement keys.

```
      \@for\@dtl@keya:=\@dtl@replacementkeys\do{%
```

Get column corresponding to this key.

```
        \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@keya}%
        \dtlgetentryfromrow{\@dtl@a}{\@dtl@col}{#1}%
```

Has value for row A been defined?

```
      \ifx\@dtl@a\dtlnovalue
```

Value for row A hasn't been defined so set to null

```
       \@dtl@setnull{\@dtl@a}{\@dtl@key}%
      \fi
```

Is value for row A null? If not null end the loop.

```
      \DTLifnull{\@dtl@a}{}{\@endfortrue}%
     }%
```

No non-null value found.

```
     \ifx\@dtl@keya\@nnil
      \let\@dtl@keya\@dtl@key
      \@dtl@setnull{\@dtl@a}{\@dtl@key}%
     \fi
    }%
    {}%
```

Check if value for row B is null.

```
    \DTLifnull{\@dtl@b}%
    {%
```

Value for row B is null, so find the first non null key in list of replacement keys.

```
     \@for\@dtl@keyb:=\@dtl@replacementkeys\do{%
```

Get column corresponding to this key.

```
      \@sdtl@getcolumnindex{\@dtl@col}{\@dtl@dbname}{\@dtl@keyb}%
      \dtlgetentryfromrow{\@dtl@b}{\@dtl@col}{#2}%
```

Has value for row B been defined?

```
      \ifx\@dtl@b\dtlnovalue
```

Value for row B hasn't been defined so set to null.

```
       \@dtl@setnull{\@dtl@b}{\@dtl@key}%
      \fi
```

Is value for row B null? If not null end the loop.

```
      \DTLifnull{\@dtl@b}{}{\@endfortrue}%
     }%
```

No non-null value found.

```
     \ifx\@dtl@keyb\@nnil
      \let\@dtl@keyb\@dtl@key
      \@dtl@setnull{\@dtl@b}{\@dtl@key}%
     \fi
    }%
    {}%
```

Compare rows A and B. First store the values for row A and B in token registers so that they can be passed to `\dtl@compare@`.

```
    \@dtl@toksA=\expandafter{\@dtl@a}%
    \@dtl@toksB=\expandafter{\@dtl@b}%
```

Do comparison.

```
      \edef\@dtl@docompare{\noexpand\dtl@compare@
        {\@dtl@keya}{\@dtl@keyb}%
        {\noexpand\@dtl@toksA}{\noexpand\@dtl@toksB}}%
      \@dtl@docompare
```

Repeat if the two values are considered identical and there are further sorting options.

```
      \ifnum\dtl@sortresult=0\relax
```

Reset switch to prevent breaking out of outer loop.

```
        \@endforfalse
      \else
```

Break out of loop.

```
        \@endfortrue
      \fi
    }%
```

Apply sort direction

```
    \multiply\dtl@sortresult by -\@dtl@sortdirection\relax
  }
```

`etsortdirection` Get the direction from either ⟨*key*⟩ or ⟨*key*⟩=⟨*direction*⟩. Sets `\@dtl@sortdirection` to either -1 (ascending) or 1 (descending).

```
  \def\@dtl@getsortdirection#1=#2\relax{%
```

Store key in `\@dtl@key`.

```
    \def\@dtl@key{#1}%
```

Store sort direction. This will be empty if no direction was specified.

```
    \def\@dtl@sortdirection{#2}%
```

Check if a direction was specified.

```
    \ifdefempty{\@dtl@sortdirection}%
    {%
```

No direction specified so assume ascending.

```
      \def\@dtl@sortdirection{-1}%
    }%
    {%
```

Get the sort direction from the second argument (needs terminating equal sign removed) and store in `\@dtl@sortdirection`.

```
      \@dtl@get@sortdirection#2%
```

Determine the direction.

```
      \def\@dtl@dir{ascending}%
      \ifx\@dtl@sortdirection\@dtl@dir
```

Ascending

```
        \def\@dtl@sortdirection{-1}%
      \else
```

Check if descending.

```
        \def\@dtl@dir{descending}%
        \ifx\@dtl@sortdirection\@dtl@dir
```

Descending

```
          \def\@dtl@sortdirection{1}%
        \else
```

Direction not valid. Generate error message.

```
          \PackageError{datatool}{Invalid sort direction
          `\@dtl@sortdirection'}{The sort direction can only be
           one of `ascending' or `descending'}%
```

Assume ascending.

```
          \def\@dtl@sortdirection{-1}%
        \fi
      \fi
    }%
  }
```

t@sortdirection Get direction (trims trailing = sign)

```
  \def\@dtl@get@sortdirection#1={\def\@dtl@sortdirection{#1}}
```

\@dtl@toksA

```
  \newtoks\@dtl@toksA
```

\@dtl@toksB

```
  \newtoks\@dtl@toksB
```

\dtl@compare

`\dtl@compare{⟨key⟩}{⟨a toks⟩}{⟨b toks⟩}`

Compares two values according to ⟨*key*⟩ of database given by `\@dtl@dbname`. Sets `\dtl@sortresult`. `\@dtl@comparecs` must be set to the required comparison macro.

```
  \newcommand{\dtl@compare}[3]{%
    \dtl@compare@{#1}{#1}{#2}{#3}%
  }
```

\dtl@compare@

`\dtl@compare@{⟨keyA⟩}{⟨keyB⟩}{⟨A toks⟩}{⟨B toks⟩}`

Compare ⟨*A*⟩ and ⟨*B*⟩ according ⟨*keyA*⟩ and ⟨*keyB*⟩ for database given by `\@dtl@dbname`. Sets `\dtl@sortresult`. `\@dtl@comparecs` must be set before use.

```
  \newcommand{\dtl@compare@}[4]{%
```

Get the data type for first key and store in `\@dtl@typeA`.

```
    \DTLgetdatatype{\@dtl@typeA}{\@dtl@dbname}{#1}%
```

Is it unset? If so, assume string

```
\ifx\@dtl@typeA\DTLunsettype
  \let\@dtl@typeA\DTLstringtype
\fi
```

Get the data type for the second key and store in `\@dtl@typeB`

```
\DTLgetdatatype{\@dtl@typeB}{\@dtl@dbname}{#2}%
```

Is it unset? If so, assume string

```
\ifx\@dtl@typeB\DTLunsettype
  \let\@dtl@typeB\DTLstringtype
\fi
```

Multiply the two values together

```
\@dtl@tmpcount=\@dtl@typeA\relax
\multiply\@dtl@tmpcount by \@dtl@typeB\relax
```

If either type is 0 (a string) then the product will also be 0 (string) otherwise it will be one of the numerical types.

```
\ifnum\@dtl@tmpcount=0\relax
```

A string, so use comparison function

```
  \edef\@dtl@tmpcmp{%
    \noexpand\@dtl@comparecs{\noexpand\dtl@sortresult}%
      {\the#3}{\the#4}%
    }%
  \@dtl@tmpcmp
  \ifdtlverbose
    \edef\@dtl@a{\the#3}%
    \edef\@dtl@b{\the#4}%
  \fi
\else
```

Store the first value

```
  \edef\@dtl@a{\the#3}%
```

Store the second value

```
  \edef\@dtl@b{\the#4}%
```

Compare

```
  \DTLifnumlt{\@dtl@a}{\@dtl@b}%
  {%
```

$A < B$

```
    \dtl@sortresult=-1\relax
  }%
  {%
    \DTLifnumgt{\@dtl@a}{\@dtl@b}%
    {%
```

$A > B$

```
      \dtl@sortresult=1\relax
    }%
    {%
```

$A = B$

```
          \dtl@sortresult=0\relax
        }%
      }%
    \fi
```

Write comparison result to terminal/log if verbose mode.

```
    \ifdtlverbose
      \@onelevel@sanitize\@dtl@a
      \@onelevel@sanitize\@dtl@b
      \dtl@message{‘\@dtl@a’ <=> ‘\@dtl@b’ = \number\dtl@sortresult}%
    \fi
  }
```

4.11 Saving a database to an external file

\@dtl@write

```
  \newwrite\@dtl@write
```

\DTLsavedb

\DTLsavedb{⟨*db name*⟩}{⟨*filename*⟩}

Save a database as an ASCII data file using the separator and delimiter given by `\@dtl@separator` and `\@dtl@delimiter`.

```
  \newcommand*{\DTLsavedb}[2]{%
    \DTLifdbexists{#1}%
    {%
```

Open output file

```
      \openout\@dtl@write=#2\relax
```

Initialise header row

```
      \def\@dtl@header{}%
```

Construct the header row

```
      \dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)%
        \in{#1}\do
      {%
        \IfSubStringInString{\@dtl@separator}{\@dtl@key}%
        {%
          \ifdefempty{\@dtl@header}%
          {%
            \protected@edef\@dtl@header{%
              \@dtl@delimiter\@dtl@key\@dtl@delimiter}%
          }%
          {%
            \toks@=\expandafter{\@dtl@header}%
            \protected@edef\@dtl@header{%
              \the\toks@\@dtl@separator
              \@dtl@delimiter\@dtl@key\@dtl@delimiter}%
          }%
        }%
        {%
          \ifdefempty{\@dtl@header}%
          {%
            \protected@edef\@dtl@header{\@dtl@key}%
          }%
          {%
            \toks@=\expandafter{\@dtl@header}%
            \protected@edef\@dtl@header{\the\toks@
              \@dtl@separator\@dtl@key}%
          }%
        }%
      }%
```

Print header

```
    \protected@write\@dtl@write{}{\@dtl@header}%
```

Iterate through each row

```
    \@sDTLforeach{#1}{}%
    {%
```

Initialise row

```
      \def\@dtl@row{}%
```

Iterate through each key

```
      \DTLforeachkeyinrow{\@dtl@val}%
      {%
        \IfSubStringInString{\@dtl@separator}{\@dtl@val}%
        {%
          \ifdefempty{\@dtl@row}%
          {%
            \protected@edef\@dtl@row{%
              \@dtl@delimiter\@dtl@val\@dtl@delimiter}%
          }%
          {%
            \toks@=\expandafter{\@dtl@row}%
            \protected@edef\@dtl@row{\the\toks@\@dtl@separator
              \@dtl@delimiter\@dtl@val\@dtl@delimiter}%
          }%
        }%
        {%
          \ifdefempty{\@dtl@row}%
          {%
            \protected@edef\@dtl@row{\@dtl@val}%
          }%
          {%
            \toks@=\expandafter{\@dtl@row}%
            \protected@edef\@dtl@row{\the\toks@\@dtl@separator
```

```
                \@dtl@val}%
            }%
          }%
        }%
```

Print row

```
          \protected@write\@dtl@write{}{\@dtl@row}%
      }%
```

Close output file

```
      \closeout\@dtl@write
    }%
    {%
      \PackageError{datatool}{Can't save database '#1': no such
        database}{}%
    }%
  }
```

\DTLsavetexdb

```
\DTLsavetexdb{⟨db name⟩}{⟨filename⟩}
```

Save a database as a LaTeX file.

```
  \newcommand*{\DTLsavetexdb}[2]{%
    \DTLifdbexists{#1}%
    {%
```

Open output file

```
      \openout\@dtl@write=#2\relax
```

Write new data base definition

```
      \protected@write\@dtl@write{}{\string\DTLnewdb{#1}}%
```

Iterate through each row

```
      \@sDTLforeach{#1}{}%
      {%
```

Start new row

```
        \protected@write\@dtl@write{}{\string\DTLnewrow*{#1}}%
```

Iterate through each column

```
        \DTLforeachkeyinrow{\@dtl@val}%
        {%
```

Is this entry null?

```
          \DTLifnull{\@dtl@val}%
          {\def\@dtl@val{}}%
          {}%
```

Add entry

```
          \protected@write\@dtl@write{}{%
            \string\DTLnewdbentry*{#1}{\dtlkey}{\@dtl@val}}%
```

```
      }%
    }%
```

Save the column headers.

```
    \dtlforeachkey(\@dtl@k,\@dtl@c,\@dtl@t,\@dtl@h)\in{#1}\do
    {%
      \@onelevel@sanitize\@dtl@h
      \protected@write\@dtl@write{}{%
        \string\DTLsetheader*{#1}{\@dtl@k}{\@dtl@h}}%
    }%
```

Store name of database in case required after database loaded:

```
    \protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
```

Close output file

```
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database '#1': no such
      database}{}%
  }%
}
```

l@saverawdbhook Hook used by \DTLsaverawdb.

```
\newcommand*{\dtl@saverawdbhook}{}
```

\DTLsaverawdb Saves given database in its internal form. Not easy for a human to read, but much faster to load.

```
\newcommand*{\DTLsaverawdb}[2]{%
  \DTLifdbexists{#1}%
  {%
```

Open output file

```
    \openout\@dtl@write=#2\relax
```

Add code at the start of the output file to check for the existence of the database:

```
    \protected@write{\@dtl@write}{}{%
      \string\DTLifdbexists{#1}\expandafter\@gobble\string\%^^J%
      {%
         \string\PackageError{datatool}{Database '#1' ^^Jalready exists}{}%
         \expandafter\@gobble\string\%^^J%
         \string\aftergroup\string\endinput
      }%
      {%
      }\expandafter\@gobble\string\%
    }%
```

Scope need to localise definitions:

```
    {%
```

```
      \def\db@row@elt@w{\expandafter\@gobble\string\%^^J\string\db@row@elt@w\space}%
      \def\db@row@elt@end@{\expandafter\@gobble\string\%^^J\string\db@row@elt@end@\space}%
      \def\db@row@id@w{\expandafter\@gobble\string\%^^J\string\db@row@id@w\space}%
      \def\db@row@id@end@{\expandafter\@gobble\string\%^^J\string\db@row@id@end@\space}%
      \def\db@col@elt@w{\expandafter\@gobble\string\%^^J\string\db@col@elt@w\space}%
      \def\db@col@elt@end@{\expandafter\@gobble\string\%^^J\string\db@col@elt@end@\space}%
      \def\db@col@id@w{\expandafter\@gobble\string\%^^J\string\db@col@id@w\space}%
      \def\db@col@id@end@{\expandafter\@gobble\string\%^^J\string\db@col@id@end@\space}%
%
      \def\db@plist@elt@w{\expandafter\@gobble\string\%^^J\string\db@plist@elt@w\space}%
      \def\db@plist@elt@end@{\expandafter\@gobble\string\%^^J\string\db@plist@elt@end@\space}%
      \def\db@key@id@w{\expandafter\@gobble\string\%^^J\string\db@key@id@w\space}%
      \def\db@key@id@end@{\expandafter\@gobble\string\%^^J\string\db@key@id@end@\space}%
      \def\db@type@id@w{\expandafter\@gobble\string\%^^J\string\db@type@id@w\space}%
      \def\db@type@id@end@{\expandafter\@gobble\string\%^^J\string\db@type@id@end@\space}%
      \def\db@header@id@w{\expandafter\@gobble\string\%^^J\string\db@header@id@w\space}%
      \def\db@header@id@end@{\expandafter\@gobble\string\%^^J\string\db@header@id@end@\space}%
```

Need to ensure the @ character can be used so \makeatletter is required, but localise the effect.

```
\protected@write{\@dtl@write}{}{\string\bgroup\string\makeatletter}%
```

If in verbose mode, add a message to let the user know what's happening when the file is later loaded.

```
\protected@write{\@dtl@write}{}{%
  \string\dtl@message{Reconstructing database^^J‘#1’}%
  \expandafter\@gobble\string\%}%
```

Save the contents of the token register that holds the column information (column id, header, type). (The write is delayed, so the contents are first expanded and stored in a temporary (global) macro to ensure its in the correct format when the write happens.)

```
\protected@write{\@dtl@write}{}{%
 \string\expandafter
 \string\global\string\expandafter^^J\string\newtoks
 \string\csname\space dtlkeys@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{%
  \string\expandafter
  \string\global^^J
  \string\csname\space dtlkeys@#1\string\endcsname
 =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
\expandafter\protected@xdef\csname dtl@rawwritedbkeys@#1\endcsname{%
  \the\csname dtlkeys@#1\endcsname}%
\protected@write{\@dtl@write}{}{\csname dtl@rawwritedbkeys@#1\endcsname}%
\protected@write{\@dtl@write}{}%
   {\expandafter\@gobble\string\}\expandafter\@gobble\string\%}%
```

Hook used by datagidx:

```
\dtl@saverawdbhook
```

Save the contents of the token register that holds the database body.

```
\protected@write{\@dtl@write}{}{%
```

```
 \string\expandafter\string\global
 \string\expandafter^^J\string\newtoks
   \string\csname\space dtldb@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{%
  \string\expandafter
  \string\global^^J\string\csname\space dtldb@#1\string\endcsname
 =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
\expandafter\protected@xdef\csname dtl@rawwritedb@#1\endcsname{\the\csname dtldb@#1\endcs
\protected@write{\@dtl@write}{}{\csname dtl@rawwritedb@#1\endcsname}%
\protected@write{\@dtl@write}{}{\expandafter\@gobble\string\}\expandafter\@gobble\string\
```

Now for the count register that keeps track of the row count.

```
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\expandafter\string\newcount
 \string\csname\space dtlrows@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\csname\space dtlrows@#1\string\endcsname
  =\expandafter\number\csname dtlrows@#1\endcsname\string\relax}%
```

Similarly for the column count.

```
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\expandafter\string\newcount
 \string\csname\space dtlcols@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\csname\space dtlcols@#1\string\endcsname
  =\expandafter\number\csname dtlcols@#1\endcsname\string\relax}%
```

Add key mappings

```
\dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
{%
  \edef\dtl@tmp{%
    \string\expandafter^^J
    \string\gdef
    \string\csname\space dtl@ci@#1@\@dtl@key\string\endcsname
    {\csname dtl@ci@#1@\@dtl@key\endcsname}\expandafter\@gobble\string\%
  }%
  \expandafter\write\expandafter\@dtl@write\expandafter{\dtl@tmp}%
}%
```

End the scope for `\makeatletter`:

```
\protected@write{\@dtl@write}{}{\string\egroup}%
```

End current scope:

```
}%
```

Store name of database in case required after database loaded:

```
\protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
```

Close output file

```
  \closeout\@dtl@write
}%
{%
```

```
    \PackageError{datatool}{Can’t save database ‘#1’: no such
      database}{}%
  }%
}
```

tectedsaverawdb Like \DTLsaverawdb but works with fragile contents. If there’s a problem with unwanted line breaks every 80 characters, try loading morewrites before datatool.

```
\newcommand*{\DTLprotectedsaverawdb}[2]{%
  \DTLifdbexists{#1}%
  {%
```

Open output file

```
    \openout\@dtl@write=#2\relax
```

Add code at the start of the output file to check for the existence of the database:

```
    \protected@write{\@dtl@write}{}{%
      \string\DTLifdbexists{#1}\expandafter\@gobble\string\%^^J%
      {%
         \string\PackageError{datatool}{Database ‘#1’ ^^Jalready exists}{}%
         \expandafter\@gobble\string\%^^J%
         \string\aftergroup\string\endinput
      }%
      {%
      }\expandafter\@gobble\string\%
    }%
```

Scope needed to localise definitions:

```
    {%
```

Need to ensure the @ character can be used so \makeatletter is required, but localise the effect.

```
      \protected@write{\@dtl@write}{}{\string\bgroup\string\makeatletter}%
```

If in verbose mode, add a message to let the user know what’s happening when the file is later loaded.

```
      \protected@write{\@dtl@write}{}{\string\dtl@message{Reconstructing database
        ^^J‘#1’}\expandafter\@gobble\string\%}%
```

Start writing the header token definition.

```
      \protected@write{\@dtl@write}{}{%
       \string\expandafter
       \string\global\string\expandafter^^J\string\newtoks
       \string\csname\space dtlkeys@#1\string\endcsname}%
      \protected@write{\@dtl@write}{}{%
        \string\expandafter
        \string\global^^J
        \string\csname\space dtlkeys@#1\string\endcsname
       =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
% Store the contents of the token register that holds the column
% information (column id, header, type) and sanitize.
%    \begin{macrocode}
```

```
\edef\dtl@rawwrite@keys{\the\csname dtlkeys@#1\endcsname}%
\@onelevel@sanitize\dtl@rawwrite@keys
```

The write can get delayed, so expand after to ensure it has the actual contents of the database rather than `\dtl@rawwrite@keys`, which may have changed by the time the write occurs. Include the closing brace of the token contents.

```
\expandafter\write\expandafter\@dtl@write\expandafter
   {\dtl@rawwrite@keys\expandafter\@gobble\string\}}%
```

Similarly for the token register that holds the database body.

```
\protected@write{\@dtl@write}{}{%
 \string\expandafter\string\global
 \string\expandafter^^J\string\newtoks
   \string\csname\space dtldb@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{%
  \string\expandafter
  \string\global^^J\string\csname\space dtldb@#1\string\endcsname
 =\expandafter\@gobble\string\{\expandafter\@gobble\string\%}%
```

```
\edef\dtl@rawwrite@db{\the\csname dtldb@#1\endcsname}%
\@onelevel@sanitize\dtl@rawwrite@db
```

Now write the sanitize contents.

```
\expandafter\write\expandafter\@dtl@write\expandafter
   {\dtl@rawwrite@db\expandafter\@gobble\string\}}%
```

Now for the count register that keeps track of the row count.

```
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\expandafter\string\newcount
 \string\csname\space dtlrows@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\csname\space dtlrows@#1\string\endcsname
  =\expandafter\number\csname dtlrows@#1\endcsname\string\relax}%
```

Similarly for the column count.

```
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\expandafter\string\newcount
 \string\csname\space dtlcols@#1\string\endcsname}%
\protected@write{\@dtl@write}{}{\string\expandafter\string\global^^J
 \string\csname\space dtlcols@#1\string\endcsname
  =\expandafter\number\csname dtlcols@#1\endcsname\string\relax}%
```

Add key mappings

```
\dtlforeachkey(\@dtl@key,\@dtl@col,\@dtl@type,\@dtl@head)\in{#1}\do
{%
  \edef\dtl@tmp{%
    \string\expandafter^^J
    \string\gdef
    \string\csname\space dtl@ci@#1@\@dtl@key\string\endcsname
    {\csname dtl@ci@#1@\@dtl@key\endcsname}\expandafter\@gobble\string\%
  }%
  \expandafter\write\expandafter\@dtl@write\expandafter{\dtl@tmp}%
```

```
    }%
```

End the scope for `\makeatletter`:

```
      \protected@write{\@dtl@write}{}{\string\egroup}%
```

End current scope:

```
    }%
```

Provide a means to keep track of the last loaded file:

```
    \protected@write{\@dtl@write}{}{\string\def\string\dtllastloadeddb{#1}}%
```

Close output file

```
    \closeout\@dtl@write
  }%
  {%
    \PackageError{datatool}{Can't save database '#1': no such
      database}{}%
  }%
}
```

4.12 Loading a database from an external file

\DTLloaddbtex

`\DTLloaddbtex{⟨cs⟩}{⟨file⟩}`

Load a `.dbtex` file and assign the database name to the control sequence ⟨*cs*⟩. Checks the file name exists and the control sequence doesn't exist.

```
\newcommand*{\DTLloaddbtex}[2]{%
  \IfFileExists{#2}%
  {%
    \input{#2}%
    \ifdef#1%
    {%
      \PackageError{datatool}{Command \string#1\space is already defined}%
      {}%
    }%
    {%
      \let#1\dtllastloadeddb
    }%
  }%
  {%
    \PackageError{datatool}{File '#2' doesn't exist.}{}%
  }%
}
```

\@dtl@read

```
\newread\@dtl@read
```

\dtl@entrycr Keep track of current column in data file

```
\newcount\dtl@entrycr
```

\ifdtlnoheader The noheader option indicates that the file doesn't have a header row.

```
\define@boolkey{loaddb}[dtl]{noheader}[true]{}
```

\ifdtlautokeys Assign the default keys even if a header row is supplied.

```
\define@boolkey{loaddb}[dtl]{autokeys}[true]{}
\dtlautokeysfalse
```

The keys option specifies the list of keys in the same order as the columns in the data file. Each key is stored in \@dtl@inky@⟨*n*⟩ where ⟨*n*⟩ is the roman numeral representation of the current column.

```
\define@key{loaddb}{keys}{%
  \dtl@entrycr=0\relax
  \@for\@dtl@key:=#1\do
  {%
    \advance\dtl@entrycr by 1\relax
    \expandafter
      \edef\csname @dtl@inky@\romannumeral\dtl@entrycr\endcsname{%
        \@dtl@key}%
  }%
}
```

The headers option specifies the list of headers in the same order as the columns in the data file.

```
\define@key{loaddb}{headers}{%
  \dtl@entrycr=0\relax
  \@for\@dtl@head:=#1\do
  {%
    \advance\dtl@entrycr by 1\relax
    \toks@=\expandafter{\@dtl@head}%
    \expandafter
      \edef\csname @dtl@inhd@\romannumeral\dtl@entrycr\endcsname{%
        \the\toks@}%
  }%
}
```

The following is supplied in a patch by Bruno Le Floch:

```
\newcount{\dtl@omitlines}
\define@key{loaddb}{omitlines}{\dtl@omitlines=#1\relax}
```

\dtldefaultkey Default key to use if none specified (column index will be appended).

```
\newcommand*{\dtldefaultkey}{Column}
```

\@dtl@readline

```
\@dtl@readline{⟨file reg⟩}{⟨cs⟩}
```

Reads line from ⟨*file reg*⟩, trims end of line character and stores in ⟨*cs*⟩.

```
\newcommand*{\@dtl@readline}[2]{%
```

Read a line from #1 and store in #2 but make sure end of line character is removed.

```
  \begingroup
  \catcode\endlinechar=\active%
  \global\read#1 to #2%
  \endgroup%
```

If empty the row starts with a comment

```
  \ifx#2\empty%
  \else%
    \expandafter\@dtl@stripeol#2%
    \let#2\@dtl@strippedline%
  \fi%
}
```

\@dtl@stripeol

```
\begingroup
\catcode\endlinechar=\active%
\gdef\@dtl@stripeol#1
{\gdef\@dtl@strippedline{#1}}
\endgroup
```

@dtl@readrawline

```
\@dtl@readrawline{⟨file register⟩}{⟨cs⟩}
```

Reads line from ⟨*file register*⟩, trims end of line character, applies mappings and stores in ⟨*cs*⟩.

```
\newcommand*{\@dtl@readrawline}[2]{%
```

Read a line from #1 and store in #2

```
  \@dtl@rawread#1 to #2%
```

Apply mappings

```
  \dtl@domappings\@dtl@line
}
```

fDTLnewdbonload

Governs whether or not the database should be defined by `\DTLloaddb` and `\DTLloadrawdb`.

```
\newif\ifDTLnewdbonload
```

Ensure compatibility with previous versions:

```
\DTLnewdbonloadtrue
```

\DTLloaddb

```
\DTLloaddb[⟨options⟩]{⟨db name⟩}{⟨filename⟩}
```

Creates a new database called ⟨*db name*⟩, and loads the data in ⟨*filename*⟩ into it. The separator and delimiter used in the file must match `\@dtl@separator` and `\@dtl@delimiter`. The optional argument is a comma-separated list.

```
\newcommand*{\DTLloaddb}{%
  \let\@dtl@doreadline\@dtl@readline
  \@dtlloaddb
}
```

`\@dtlloaddb` Loads database using `\@dtl@doreadline` to read and trim line from file. (`\@dtl@doreadline` must be set before use.)

```
\newcommand*{\@dtlloaddb}[3][]{%
```

Check if file exists

```
    \IfFileExists{#3}{%
```

File exists. Locally change catcode of double quote character in case it has been made active.

```
    \begingroup
      \catcode`\"12\relax
```

Initialise default options

```
       \dtlnoheaderfalse
```

Get the options

```
       \setkeys{loaddb}{#1}%
```

Open the file for reading.

```
      \openin\@dtl@read=#3%
      \dtl@message{Reading `#3'}%
```

The following supplied in patch by Bruno Le Floch:

```
      \loop
      \ifnum \dtl@omitlines > \z@
        \advance\dtl@omitlines by \m@ne
        \read\@dtl@read to \@dtl@line
      \repeat
```

Create the database if required.

```
      \ifDTLnewdbonload
       \DTLnewdb{#2}%
      \fi
```

Check if the file is empty.

```
      \ifeof\@dtl@read
```

File is empty, so just issue a warning.

```
        \PackageWarning{datatool}{File `#3' has no data}%
      \else
```

Does the file have a header row?

```
        \ifdtlnoheader
        \else
```

Remove initial blank rows

```
\loop
```

Set repeat condition to false

```
  \@dtl@conditionfalse
```

Do nothing if reached the end of file

```
  \ifeof\@dtl@read
  \else
```

Read a line from the file and store in \@dtl@line

```
   \@dtl@doreadline\@dtl@read\@dtl@line
```

If this is a blank row, set repeat condition to true

```
    \ifdefempty{\@dtl@line}%
    {%
      \@dtl@conditiontrue
    }%
    {%
    }%
  \fi
```

Repeat loop if necessary

```
\if@dtl@condition
\repeat
```

Parse the header row. Store the row as ⟨*sep*⟩⟨*row*⟩⟨*sep*⟩ in \@dtl@lin@.

```
\protected@edef\@dtl@lin@{%
   \@dtl@separator\@dtl@line\@dtl@separator}%
```

Keep track of columns:

```
\dtl@entrycr=0\relax
```

Keep lopping off elements until the end of the row is reached. (That is, until \@dtl@lin@ is \@dtl@separator.)

```
\loop
```

Lopoff the first element and store in \@dtl@key

```
\expandafter\@dtl@lopoff\@dtl@lin@\to\@dtl@lin@\@dtl@key
```

Increment column count.

```
\advance\dtl@entrycr by 1\relax
```

If autokeys option is on, add generic key

```
\ifdtlautokeys
  \csedef{@dtl@inky@\romannumeral\dtl@entrycr}%
     {\dtldefaultkey\number\dtl@entrycr}%
\else
```

If missing a key, add generic one:

```
  \ifdefempty{\@dtl@key}%
  {%
    \edef\@dtl@key{\dtldefaultkey\number\dtl@entrycr}%
  }%
  {}%
\fi
```

Store key in `\@dtl@toks`

```
\expandafter\@dtl@toks\expandafter{\@dtl@key}%
```

Store the key in `\@dtl@inky@`⟨*n*⟩ where ⟨*n*⟩ is the roman numeral representation of the current column, unless already defined.

```
\@ifundefined{@dtl@inky@\romannumeral\dtl@entrycr}%
{%
  \expandafter
    \edef\csname @dtl@inky@\romannumeral
      \dtl@entrycr\endcsname{\the\@dtl@toks}%
}%
{%
```

If key has been specified in `#1`, then use the header found in the file, unless a header has also been specified in `#1`

```
  \@ifundefined{@dtl@inhd@\romannumeral\dtl@entrycr}%
  {%
    \expandafter
      \edef\csname @dtl@inhd@\romannumeral
        \dtl@entrycr\endcsname{\the\@dtl@toks}%
  }%
  {}%
}%
```

Check if the loop should be repeated

```
\ifx\@dtl@lin@\@dtl@separator
  \@dtl@conditionfalse
\else
  \@dtl@conditiontrue
\fi
```

Repeat loop if necessary.

```
\if@dtl@condition
\repeat
```

End if no header

```
\fi
```

Now for the rest of the data. If the end of file has been reached, then only the header row is available or file is empty.

```
\ifeof\@dtl@read
   \ifdtlnoheader
     \PackageWarning{datatool}{No data in ‘#3’}%
   \else
     \PackageWarning{datatool}{Only header row found in ‘#3’}%
   \fi
\else
```

Iterate through the rest of the file. First set the repeat condition to true:

```
\@dtl@conditiontrue
\loop
```

Read in a line

```
 \@dtl@doreadline\@dtl@read\@dtl@line
```

Check if the line is empty.

```
  \ifdefempty{\@dtl@line}%
  {%
```

Do nothing if the row is empty.

```
  }%
  {%
```

Add a new row to the database. (Don't need to check if the database exists, since it's just been created.)

```
    \@sDTLnewrow{#2}%
```

Store the row as ⟨*sep*⟩⟨*row*⟩⟨*sep*⟩ to make the lopping off easier

```
    \expandafter\@dtl@toks\expandafter{\@dtl@line}%
    \edef\@dtl@lin@{\@dtl@separator\the\@dtl@toks
      \@dtl@separator}%
```

Reset the column counter.

```
    \dtl@entrycr=0\relax
```

Iterate through each element in the row. Needs to be grouped since we're already inside a loop.

```
    {%
```

Initialise repeat condition

```
      \@dtl@conditiontrue
```

Iterate through the list

```
      \loop
```

lop off first element and store in `\@dtl@thisentry`

```
        \expandafter\@dtl@lopoff\@dtl@lin@\to
          \@dtl@lin@\@dtl@thisentry
```

Increment the column count.

```
        \advance\dtl@entrycr by 1\relax
```

Get the key for this column and store in `\@dtl@thiskey`. Use default value if not defined.

```
        \@ifundefined{@dtl@inky@\romannumeral\dtl@entrycr}%
        {%
          \edef\@dtl@thiskey{\dtldefaultkey
            \number\dtl@entrycr}%
          \expandafter\let
            \csname @dtl@inky@\romannumeral
              \dtl@entrycr\endcsname\@dtl@thiskey
        }%
```

```
             {%
               \edef\@dtl@thiskey{%
                 \csname @dtl@inky@\romannumeral
                   \dtl@entrycr\endcsname}%
             }%
```

Store this entry in `\@dtl@toks`

```
             \expandafter\@dtl@toks\expandafter{\@dtl@thisentry}%
```

Add this entry to the database

```
             \edef\@do@dtlnewentry{\noexpand\@sDTLnewdbentry
               {#2}{\@dtl@thiskey}{\the\@dtl@toks}}%
             \@do@dtlnewentry
```

Check if loop should be terminated

```
             \ifx\@dtl@lin@\@dtl@separator
               \@dtl@conditionfalse
             \fi
```

Repeat loop if necessary

```
           \if@dtl@condition
           \repeat
         }%
```

End of parsing this row

```
     }%
```

If the end of file has been reached, set the repeat condition to false.

```
     \ifeof\@dtl@read \@dtl@conditionfalse\fi
```

Repeat if necessary

```
     \if@dtl@condition
     \repeat
   \fi
```

End of first `\ifeof`

```
\fi
```

Close the input file

```
\closein\@dtl@read
```

Set the headers if required

```
 \edef\@dtl@maxcols{\expandafter
   \number\csname dtlcols@#2\endcsname}%
 \dtlgforint\dtl@entrycr=1\to\@dtl@maxcols\step1\do
 {%
   \@ifundefined{@dtl@inhd@\romannumeral\dtl@entrycr}%
   {}%
   {%
      \expandafter\let\expandafter\@dtl@head
        \csname @dtl@inhd@\romannumeral\dtl@entrycr\endcsname
      \@dtl@toks=\expandafter{\@dtl@head}%
      \edef\@dtl@dosetheader{\noexpand\@dtl@setheaderforindex
```

```
            {#2}{\number\dtl@entrycr}{\the\@dtl@toks}}%
          \@dtl@dosetheader
       }%
     }%
```

End current scope

```
  \endgroup
```

End true part of if file exists

```
  }{%
```

Requested file not found on TeX’s path

```
    \PackageError{datatool}{Can’t load database ‘#2’ (file ‘#3’
    doesn’t exist)}{}%
  }%
}
```

\DTLloadrawdb

\DTLloadrawdb{⟨*db name*⟩}{⟨*filename*⟩}

Loads a raw database (substitutes % → \%, $ → \$, & → \&, # → \#, ~ → \textasciitilde, _ → _ and ^ → \textasciicircum.) The user can add additional mappings.

```
\newcommand*\DTLloadrawdb{%
  \let\@dtl@doreadline\@dtl@readrawline
  \@dtlloaddb
}
```

\@dtl@rawread

\@dtl@rawread⟨*number*⟩to⟨*cmd*⟩

Reads in a raw line from file given by ⟨*number*⟩ converts special characters and stores in ⟨*cmd*⟩

```
\begingroup
\catcode‘\%=\active
\catcode‘$=\active
\catcode‘&=\active
\catcode‘~=\active
\catcode‘_=\active
\catcode‘^=\active
\catcode‘#=\active
\catcode‘?=6\relax
\catcode‘<=1\relax
\catcode‘>=2\relax
\catcode‘\{=\active
\catcode‘\}=\active
\gdef\@dtl@rawread?1to?2<\relax
<<\catcode‘\%=\active
```

```
\catcode`$=\active
\catcode`&=\active
\catcode`~=\active
\catcode`_=\active
\catcode`^=\active
\catcode`#=\active
\catcode`\{=\active
\catcode`\}=\active
\def%<\noexpand\%>\relax
\def$<\noexpand\$>\relax
\def&<\&>\relax
\def#<\#>\relax
\def~<\noexpand\textasciitilde>\relax
\def_<\noexpand\_>\relax
\def^<\noexpand\textasciicircum>\relax
\@dtl@activatebraces
\@dtl@doreadraw?1?2>>>
\gdef\@dtl@doreadraw?1?2<\relax
\begingroup\catcode\endlinechar=\active\global\read?1 to \dtl@tmp\endgroup
\expandafter\@dtl@stripeol\dtl@tmp
\let\dtl@tmp\@dtl@strippedline
\protected@xdef?2<\dtl@tmp>\relax
>
\endgroup
```

.

@activatebraces \@dtl@activatebraces resets braces for \@dtl@rawread

```
\begingroup
\catcode`\{=\active
\catcode`\}=\active
\catcode`<=1\relax
\catcode`>=2\relax
\gdef\@dtl@activatebraces<%
 \catcode`\{=\active
 \catcode`\}=\active
 \def{<\noexpand\{>%
 \def}<\noexpand\}>%
>%
\endgroup
```

\DTLrawmap

\DTLrawmap{⟨*string*⟩}{⟨*replacement*⟩}

Additional mappings to perform when reading a raw data file

```
\newcommand*{\DTLrawmap}[2]{%
  \expandafter\@dtl@toks\expandafter{\@dtl@rawmappings}%
  \ifdefempty{\@dtl@rawmappings}%
```

```
    {%
      \def\@dtl@rawmappings{{#1}{#2}}%
    }%
    {%

      \def\@dtl@tmp{{#1}{#2}}%
      \protected@edef\@dtl@rawmappings{\the\@dtl@toks,\@dtl@tmp}%
    }%
  }
```

dtl@rawmappings

List of mappings.

```
  \newcommand*{\@dtl@rawmappings}{}
```

\dtl@domappings

\dtl@domappings{⟨*cmd*⟩}

Do all mappings in string given by ⟨*cmd*⟩.

```
  \newcommand*{\dtl@domappings}[1]{%
    \@for\@dtl@map:=\@dtl@rawmappings\do{%
      \expandafter\DTLsubstituteall\expandafter#1\@dtl@map
    }%
  }
```

4.13 Debugging commands

These commands are provided to assist debugging

\dtlshowdb

\dtlshowdb{⟨*db name*⟩}

Shows the database.

```
  \newcommand*{\dtlshowdb}[1]{%
    \expandafter\showthe\csname dtldb@#1\endcsname
  }
```

\dtlshowdbkeys

\dtlshowdbkeys{⟨*db name*⟩}

Shows the key list for the named database.

```
  \newcommand*{\dtlshowdbkeys}[1]{%
    \expandafter\showthe\csname dtlkeys@#1\endcsname
  }
```

\dtlshowtype

```
\dtlshowtype{⟨db name⟩}{⟨key⟩}
```

Show the data type for given key in the named database. This should be an integer from 0 to 3.

```
\newcommand*{\dtlshowtype}[2]{%
  \DTLgetdatatype{\@dtl@type}{#1}{#2}\show\@dtl@type
}
```

5 datagidx.sty

This package provides a means to produces indices and glossaries without the need for an external indexing application, such as `makeindex` or `xindy`. However, the code here has been developed to implement the word order style described by the Oxford Style Manual. If you are not writing in English, this may not be applicable to your needs. You may be able to define your own comparison handler to use with `\dtlsort`. If not, you'll need to use `xindy` with a package such as glossaries.

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datagidx}[2019/09/27 v2.32 (NLCT)]
```

Required packages:

```
\RequirePackage{datatool}
\RequirePackage{etoolbox}
\RequirePackage{xkeyval}
\RequirePackage{mfirstuc}
\RequirePackage{xfor}
\RequirePackage{multicol}
\RequirePackage{textcase}

\RequirePackage{afterpage}
```

5.1 Default Settings

These commands need to be defined before the package options are used.

`atagidx@columns` The number of columns to use for the index/glossary.

```
\newcommand*{\datagidx@columns}{2}
```

`LgidxSetColumns`

```
\newcommand*{\DTLgidxSetColumns}[1]{%
  \DTLifint{#1}%
  {%
    \def\datagidx@columns{#1}%
  }%
  {%
    \PackageError{datagidx}%
    {Number of columns must be an integer}%
    {%
      You have requested ‘#1’ columns, which can’t be parsed as a
      number%
      }%
    }%
  }
```

LgidxChildCount Child counter.

```
  \newcounter{DTLgidxChildCount}
```

LgidxChildCount Reduce duplicate identifier warnings if hyperref in use.

```
  \def\theHDTLgidxChildCount{\Label.\arabic{DTLgidxChildCount}}
```

ChildCountLabel Label for child counter.

```
  \newcommand*{\DTLgidxChildCountLabel}{\theDTLgidxChildCount) }
```

LgidxChildStyle Should the child name be displayed? (Default: show name.) If the name shouldn't be displayed, replace with a number.

```
  \newcommand*{\DTLgidxChildStyle}[1]{#1}
```

x@setchildstyle

```
  \newcommand*{\datagidx@setchildstyle}[1]{%
    \ifcase#1\relax
      \renewcommand*{\DTLgidxChildStyle}[1]{##1}%
    \or
      \renewcommand*{\DTLgidxChildStyle}[1]{%
        \DTLgidxChildCountLabel
      }%
    \fi
  }
```

dx@foreachchild Iterate through each child label

```
  \newcommand{\datagidx@foreachchild}{%
    \datagidx@sort@foreachchild
  }
```

dx@setchildsort

```
  \newcommand*{\datagidx@setchildsort}[1]{%
    \ifcase#1\relax
      \renewcommand*{\datagidx@foreachchild}{%
        \datagidx@sort@foreachchild
      }%
    \or
      \renewcommand*{\datagidx@foreachchild}{%
        \datagidx@unsort@foreachchild
      }%
    \fi
  }
```

DTLgidxPostName What to put after the name. (Defaults to space.)

```
  \newcommand*{\DTLgidxPostName}{ }
```

dxPostChildName What to put after the child name.

```
\newcommand*{\DTLgidxPostChildName}{\DTLgidxPostName}
```

DTLgidxNameCase Should the name have a case change in the index/glossary? (Default: no change.)

```
\newcommand*{\DTLgidxNameCase}[1]{#1}
```

idx@setnamecase

```
\newcommand*{\datagidx@setnamecase}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxNameCase}[1]{##1}%
  \or
    \let\DTLgidxNameCase\MakeTextUppercase
  \or
    \let\DTLgidxNameCase\MakeTextLowercase
  \or
    \let\DTLgidxNameCase\xmakefirstuc
  \or
    \let\DTLgidxNameCase\xcapitalisewords
  \fi
}
```

DTLgidxNameFont The font to use for the name in the index/glossary. (Default: normal font.)

```
\newcommand*{\DTLgidxNameFont}[1]{\textnormal{#1}}
```

PostDescription What to put after the description. (Defaults to nothing.)

```
\newcommand*{\DTLgidxPostDescription}{}
```

idx@setpostdesc

```
\newcommand*{\datagidx@setpostdesc}[1]{%
   \ifcase#1\relax
     \renewcommand*{\DTLgidxPostDescription}{}%
   \or
     \renewcommand*{\DTLgidxPostDescription}{.}%
   \fi
 }
```

gidxPreLocation What to put before the location list. (Defaults to en-space.)

```
\newcommand*{\DTLgidxPreLocation}{\enspace}
```

@setprelocation

```
\newcommand*{\datagidx@setprelocation}[1]{%
   \ifcase#1\relax
     \renewcommand*{\DTLgidxPreLocation}{}%
   \or
     \renewcommand*{\DTLgidxPreLocation}{\enspace}%
   \or
     \renewcommand*{\DTLgidxPreLocation}{ }%
   \or
```

```
      \renewcommand*{\DTLgidxPreLocation}{\dotfill}%
    \or
      \renewcommand*{\DTLgidxPreLocation}{\hfill}%
    \fi
  }
```

DTLgidxLocation How to display the location. (Defaults to show the location list.)

```
\newcommand*{\DTLgidxLocation}{\dtldolocationlist}
```

idx@setlocation Should the location list be displayed?

```
\newcommand*{\datagidx@setlocation}[1]{%
  \ifcase#1\relax
    \renewcommand*{\DTLgidxLocation}{}%
  \or
    \renewcommand*{\DTLgidxLocation}{\dtldolocationlist}%
  \or
    \renewcommand*{\DTLgidxLocation}{\dtldofirstlocation}%
  \fi
}
```

\DTLgidxSee How to display the cross-reference list.

```
\newcommand*{\DTLgidxSee}{%
  \DTLifnull{\See}%
  {}%
  {%
    \DTLgidxPreLocation
    \DTLgidxFormatSee{\seename}{\See}%
  }%
}
```

\DTLgidxSeeAlso How to display the “see also” list.

```
\newcommand*{\DTLgidxSeeAlso}{%
  \DTLifnull{\SeeAlso}%
  {}%
  {%
     \DTLgidxFormatSeeAlso{\seealsoname}{\SeeAlso}%
  }%
}
```

ChildrenSeeAlso Display the children and the see also attributes.

```
\newcommand*{\DTLgidxChildrenSeeAlso}{%
  \DTLgidxChildren
  \DTLgidxSeeAlso
}
```

datagidx@setsee How should cross-references appear?

```
\newcommand*{\datagidx@setsee}[1]{%
  \ifcase#1\relax
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}%
    {%
      , \DTLgidxFormatSee{\seename}{\See}%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}
    {%
      \space(\DTLgidxFormatSee{\seename}{\See})%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}%
    {%
      . \DTLgidxFormatSee{\xmakefirstuc{\seename}}{\See}%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}
    {%
      \space\DTLgidxFormatSee{\seename}{\See}%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}
    {%
      \DTLgidxFormatSee{\seename}{\See}%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}
    {%
      ; \DTLgidxFormatSee{\seename}{\See}%
    }%
  }%
\or
  \renewcommand*{\DTLgidxSee}{%
    \DTLifnull{\See}{}
    {%
      \DTLgidxPreLocation\DTLgidxFormatSee{\seename}{\See}%
    }%
  }%
\fi
```

```
}
```

LgidxSymDescSep Separator character between symbol and description if both are present.

```
\newcommand*{\DTLgidxSymDescSep}{\space}
```

gidxsymbolwidth Space to allocate for the symbol. If zero or negative, symbol just occupies its natural space.

```
\newlength\datagidxsymbolwidth
```

dxlocationwidth Space to allocate for the location list. If zero or negative, the list just occupies its natural space.

```
\newlength\datagidxlocationwidth
```

LgidxFormatDesc How to format the description.

```
\newcommand{\DTLgidxFormatDesc}[1]{#1}
```

mbolDescription How to format the symbol and description fields.

```
\newcommand*{\DTLgidxSymbolDescription}{%
  \DTLgidxSymbolDescLeft
  \DTLgidxSymbolDescRight
}
\newcommand*{\DTLgidxSymbolDescLeft}{%
  \ifdefempty{\Symbol}{}{(\Symbol)\DTLgidxSymDescSep}%
}
\newcommand*{\DTLgidxSymbolDescRight}{%
  \ifdefempty{\Description}{}%
  {%
    \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
  }%
}
```

agidxsymbolleft Identifies whether the symbol has been set to left or right.

```
\newif\if@datagidxsymbolleft
\@datagidxsymbollefttrue
```

x@formatsymdesc

```
\newcommand*{\datagidx@formatsymdesc}[1]{%
   \ifcase#1\relax
```

Only symbol

```
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Symbol}{}{\Symbol}%
      }%
      \renewcommand*{\DTLgidxSymbolDescRight}{}%
      \@datagidxsymbollefttrue
   \or
```

Only description

```
      \renewcommand*{\DTLgidxSymbolDescLeft}{%
        \ifdefempty{\Description}{}%
```

```
     {%
        \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
     }%
   }%
   \renewcommand*{\DTLgidxSymbolDescRight}{}%
   \@datagidxsymbolleftfalse
\or
```

(symbol) description

```
   \renewcommand*{\DTLgidxSymbolDescLeft}{%
     \ifdefempty{\Symbol}{}{(\Symbol)\DTLgidxSymDescSep}%
   }%
   \renewcommand*{\DTLgidxSymbolDescRight}{%
     \ifdefempty{\Description}{}%
     {%
       \DTLgidxFormatDesc{\Description}\DTLgidxPostDescription
     }%
   }%
   \@datagidxsymbollefttrue
\or
```

description (symbol)

```
   \renewcommand*{\DTLgidxSymbolDescLeft}{%
     \ifdefempty{\Description}{}%
     {%
       \DTLgidxFormatDesc{\Description}%
       \DTLgidxPostDescription\DTLgidxSymDescSep
     }%
   }%
   \renewcommand*{\DTLgidxSymbolDescRight}{%
     \ifdefempty{\Symbol}{}{(\Symbol)}%
   }%
   \@datagidxsymbolleftfalse
\or
```

symbol description

```
   \renewcommand*{\DTLgidxSymbolDescLeft}{%
     \ifdefempty{\Symbol}{}{\Symbol\DTLgidxSymDescSep}%
   }%
   \renewcommand*{\DTLgidxSymbolDescRight}{%
     \ifdefempty{\Description}{}%
     {%
       \DTLgidxFormatDesc{\Description}%
       \DTLgidxPostDescription
     }%
   }%
   \@datagidxsymbollefttrue
\or
```

description symbol

```
   \renewcommand*{\DTLgidxSymbolDescLeft}{%
```

```
       \ifdefempty{\Description}{}%
       {%
         \DTLgidxFormatDesc{\Description}%
         \DTLgidxPostDescription\DTLgidxSymDescSep
       }%
     }%
     \renewcommand*{\DTLgidxSymbolDescRight}{%
       \ifdefempty{\Symbol}{}{\Symbol}%
     }%
     \@datagidxsymbolleftfalse
  \fi
}
```

idxSetCompositor

\DTLgidxSetCompositor{⟨*symbol*⟩}

Set the location compositor.

```
\newcommand*{\DTLgidxSetCompositor}[1]{%
 \undef\datagidx@docomplist
 \DeclareListParser{\datagidx@docomplist}{#1}%
 \def\datagidx@compositor{#1}%
}
```

Set the default compositor to `.` (full stop).

```
\DTLgidxSetCompositor{.}
```

Sorting can take a long time (especially with large databases) but two LaTeX runs are usually required to get the index or glossary up-to-date, so we usually don't need to worry about sorting on the first run (unless the order in some way affects the document, e.g. the group headings are to appear in the table of contents). It may also be that some modifications are done to the document that don't require a re-sort. The optimize setting tries to minimize the amount of sorting done to help speed up document compilation.

There are to optimization levels: low and high. The low level optimization just sorts every other LaTeX run. This is done by writing to the aux file to determine whether or not the sort should be done next run. This is a cheap and easy hack that won't work if sorting makes the document out-of-date (for example, if the sorted index or glossary affects the table of contents by, say, making the group headings a sectional unit).

The high level optimization is more complicated and involves writing the sorted database to an external file and reading it in on the next run. This requires checks to see if the location lists have changed, in which case a new sort may be required.

The optimization function is only implemented when the sorting is specified via the sort key. Any explicit sorting done by the user via commands such as `\dltsort` are not effected by the optimization setting.

atagidx@do@sort Indicate what to do when it's time to sort the index/glossary. This defaults to un-optimised setting to avoid confusing users who don't like to read the manual.

```
\newcommand*{\datagidx@do@sort}{\datagidx@sort}
```

First deal with the low-level optimization as it's easier to implement.

x@optimize@sort The code to perform when the low optimize setting is on. If the command `\datagidx@do@optimize@sort` has been defined, do the sort. If it hasn't been defined, don't sort. If a sort isn't performed, the command definition is written to the aux file. If a sort is performed, the command definition isn't written to the aux file. This will do the sort every other run.

```
\newcommand*{\datagidx@optimize@sort}{%
```

First, has `\datagidx@do@optimize@sort` been defined?

```
  \ifdef\datagidx@do@optimize@sort
  {%
```

It has been defined so go ahead and do the sort.

```
    \datagidx@sort
  }%
  {%
```

It hasn't been defined so don't sort. Write the command definition into the aux file for the next run.

```
    \protected@write\@auxout{}{%
      \string\gdef\string\datagidx@do@optimize@sort{}%
    }%
```

Let the user know they need to recompile the document.

```
    \global\let\@datagidx@dorerun@warn@sort\@data@rerun@warn@sort
  }%
}
```

f@datagidx@warn Provide a switch to allow warnings to be suppressed.

```
\newif\if@datagidx@warn
\@datagidx@warntrue
```

dx@dorerun@warn

```
\newcommand*\@datagidx@dorerun@warn{}
\AtEndDocument{\if@datagidx@warn\@datagidx@dorerun@warn\fi}
```

rerun@warn@sort

```
\newcommand*\@datagidx@dorerun@warn@sort{}
\AtEndDocument{\if@datagidx@warn\@datagidx@dorerun@warn@sort\fi}
```

rerun@warn@sort Warning issued when a rerun is required to sort the index or glossary.

```
\newcommand*\@data@rerun@warn@sort{%
  \PackageWarningNoLine{datagidx}{Rerun required to sort the
   index/glossary databases}%
}
```

gidx@rerun@warn Warning issued when a rerun is required to update the location lists.

```
\newcommand*\@data@rerun@warn{%
  \PackageWarningNoLine{datagidx}{Rerun required to ensure the
   index/glossary location lists are up-to-date}%
}
```

The high optimize setting is more complicated. This involves writing each database to an external file (named `\jobname-`⟨*db label*⟩.gidx). The sort is only performed if new terms are added or used.

ighopt@optimize

```
\newcommand*{\datagidx@do@highopt@optimize}{%
  \renewcommand*{\datagidx@do@sort}{%
```

Only sort if database has changed.

```
    \ifcsdef{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
    {%
      \csuse{datagidx@do@highopt@sort@\DTLgidxCurrentdb}%
    }%
    {%
```

Do nothing

```
    }%
```

Save the database to file.

```
    \bgroup
```

Hook into write macro to clear certain fields and protect commands like `\DTLgidxName`.

```
    \def\dtl@saverawdbhook{%
      \let\db@col@id@w\@datagidx@db@col@id@w
      \def\DTLgidxName{\string\DTLgidxName\space}%
      \def\DTLgidxMac{\string\DTLgidxMac\space}%
      \def\DTLgidxRank{\string\DTLgidxRank\space}%
      \def\DTLgidxParen{\string\DTLgidxParen\space}%
      \def\DTLgidxParticle{\string\DTLgidxParticle\space}%
      \def\DTLgidxOffice{\string\DTLgidxOffice\space}%
      \def\DTLgidxSaint{\string\DTLgidxSaint\space}%
      \def\DTLgidxPlace{\string\DTLgidxPlace\space}%
      \def\DTLgidxIgnore{\string\DTLgidxIgnore\space}%
      \def\DTLgidxNameNum{\string\DTLgidxNameNum\space}%
      \def\DTLgidxSubject{\string\DTLgidxSubject\space}%
    }%
    \DTLsaverawdb{\DTLgidxCurrentdb}{\datagidxhighoptfilename\DTLgidxCurrentdb}%
    \egroup
  }%
```

Change the behaviour of `\newgidx`

```
  \def\newgidx{\datagidx@highopt@newgidx}%
```

Change the behaviour of `\newterm`

```
  \def\newterm{\datagidx@highopt@newterm}%
}
```

idx@db@col@id@w A bit of trickery is need to clear the Used and Location fields when writing the raw database to file.

```
\def\@datagidx@db@col@id@w#1\db@col@id@end@\db@col@elt@w#2\db@col@elt@end@\db@col@id@w#3\db@col
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@w\space #1%
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@end@\space
  \expandafter\@gobble\string\%^^J
  \string\db@col@elt@w\space
   \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Used\endcsname=#1\space
     0%
   \else
     \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Location\endcsname=#1\space
     \else
       \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @CurrentLocation\endcsname=#1\space
       \else
```

We also want to prevent the first character of the sort field from being expanded to help get the group correct (in case the user wants to sort on, say, the tilde character).

```
         \expandafter\ifnum\csname dtl@ci@\DTLgidxCurrentdb @Sort\endcsname=#1\space
           \protect#2%
         \else
         #2%
         \fi
       \fi
     \fi
   \fi
  \expandafter\@gobble\string\%^^J
  \string\db@col@elt@end@\space
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@w\space #3%
  \expandafter\@gobble\string\%^^J
  \string\db@col@id@end@\space
}
```

With the 'highopt optimize' setting, whenever a location is written to the aux file, if no location has been defined the database needs sorting.

@highopt@update Default does nothing. (Argument is the entry's label.)

```
\newcommand*{\datagidx@do@highopt@update}[1]{}
```

highoptfilename Expands to the name of the filename associated with the database identified by the argument for the 'highopt' setting.

```
\newcommand*{\datagidxhighoptfilename}[1]{\jobname-#1.gidx}
```

5.2 Package Options

optimize A boolean option indicating whether or not to optimize the sort. This is only available as a global option. If you want to optimize some glossaries but not others, switch on the optimize function and clear the sort key for the relevant glossaries and manually sort using \dtlsort before the glossary is displayed.

```
\define@choicekey{datagidx.sty}{optimize}[\val\nr]%
 {off,low,high}[high]%
 {%
   \ifcase\nr\relax
    \renewcommand*{\datagidx@do@sort}{\datagidx@sort}
   \or
    \renewcommand*{\datagidx@do@sort}{\datagidx@optimize@sort}
   \or
    \datagidx@do@highopt@optimize
   \fi
 }
```

nowarn A boolean option to suppress warnings.

```
\define@choicekey{datagidx.sty}{nowarn}[\val\nr]{true,false}[true]%
{%
  \ifcase\nr\relax
    \@datagidx@warnfalse
  \or
    \@datagidx@warntrue
  \fi
}
```

utf8

```
\define@choicekey{datatool.sty}{utf8}{true,false}[true]{%
  \setbool{@dtl@utf8}{#1}%
}
```

These options govern the general layout of the glossary/index.

columns The number of columns used by multicols (or multicols*). If only one column is specified, multicols (or multicols*) isn't used.

```
\define@key{datagidx.sty}{columns}%
{%
  \DTLgidxSetColumns{#1}%
}
```

child Indicates whether or not to show the name in child entries.

```
\define@choicekey{datagidx.sty}{child}[\val\nr]%
 {named,noname}%
 {%
   \datagidx@setchildstyle\nr
 }
```

namecase Options for name case.

```
\define@choicekey{datagidx.sty}{namecase}[\val\nr]%
 {nochange,uc,lc,firstuc,capitalise}%
 {%
   \datagidx@setnamecase\nr
 }
```

namefont Option to set the name font.

```
\define@key{datagidx.sty}{namefont}%
{%
  \renewcommand*{\DTLgidxNameFont}[1]{{#1{##1}}}%
}
```

postname What to put after the name.

```
\define@key{datagidx.sty}{postname}
{%
  \renewcommand*{\DTLgidxPostName}{#1}%
}
```

postdesc what to put after the description.

```
\define@choicekey{datagidx.sty}{postdesc}[\val\nr]%
 {none,dot}%
 {%
   \datagidx@setpostdesc\nr
 }
```

prelocation What to put before the location list.

```
\define@choicekey{datagidx.sty}{prelocation}[\val\nr]%
 {none,enspace,space,dotfill,hfill}%
 {%
   \datagidx@setprelocation\nr
 }
```

location How to display the location list.

```
\define@choicekey{datagidx.sty}{location}[\val\nr]%
 {hide,list,first}%
 {\datagidx@setlocation\nr}
```

see How to display the cross-reference list.

```
\define@choicekey{datagidx.sty}{see}[\val\nr]%
  {comma,brackets,dot,space,nosep,semicolon,location}%
  {\datagidx@setsee\nr}
```

symbol How to format the symbol in relation to the description.

```
\define@choicekey{datagidx.sty}{symboldesc}[\val\nr]%
 {symbol,desc,(symbol) desc,desc (symbol),symbol desc,desc symbol}%
 {\datagidx@formatsymdesc\nr}
```

compositor Location compositor.

```
\define@key{datagidx.sty}{compositor}%
{%
  \DTLgidxSetCompositor{#1}%
}%
```

final

```
\DeclareOptionX{final}{%
  \let\datagidxshowifdraft\@gobble
}
```

Set as default:

```
\let\datagidxshowifdraft\@gobble
```

draft

```
\DeclareOptionX{draft}{%
  \let\datagidxshowifdraft\@firstofone
}
```

verbose

```
\define@choicekey{datagidx.sty}{verbose}[\val\nr]%
 {true,false}[true]%
 {%
   \csuse{dtlverbose\val}%
 }
```

Process package options:

```
\ProcessOptionsX
```

Database to keep track of all the defined terms.

```
\DTLnewdb{datagidx}
```

5.3 Glossary/Index Formatting

\seename

```
\providecommand*{\seename}{see}
```

\seealsoname

```
\providecommand*{\seealsoname}{see also}
```

LgidxSeeTagFont

```
\newcommand*{\DTLgidxSeeTagFont}[1]{\emph{#1}}
```

DTLgidxFormatSee

`\DTLgidxFormatSee{⟨tag⟩}{⟨label list⟩}`

```
\newcommand*{\DTLgidxFormatSee}[2]{%
  \DTLgidxSeeTagFont{#1} \DTLgidxSeeList{#2}%
}
```

idxFormatSeeAlso

\DTLgidxFormatSeeAlso{⟨*tag*⟩}{⟨*label list*⟩}

```
\newcommand*{\DTLgidxFormatSeeAlso}[2]{%
  \datagidxdoseealso
  {%
    \DTLgidxSeeTagFont{#1} \DTLgidxSeeList{#2}%
  }%
}
```

tagidxdoseealso

```
\newcommand*{\datagidxdoseealso}[1]{%
  \datagidxseealsostart
   #1%
  \datagidxseealsoend
}
```

\DTLgidxSeeList

\DTLgidxSeeList{⟨*label list*⟩}

```
\newcommand*{\DTLgidxSeeList}[1]{%
  \def\datagidx@sep{}%
  \@for\dtl@thislabel:=#1\do
  {%
    \ifx\@xfor@nextelement\@nnil
```

Last iteration.

```
      \ifdefempty{\datagidx@sep}%
      {%
```

Only one element in the list.

```
      }%
      {%
```

Not the only element in the list.

```
        \DTLidxSeeLastSep
      }%
    \else
```

Not last iteration

```
      \datagidx@sep
      \let\datagidx@sep\DTLidxSeeSep
    \fi
    \DTLidxFormatSeeItem{\dtl@thislabel}%
  }%
}
```

idxFormatSeeItem

\DTLidxFormatSeeItem{⟨*label*⟩}

```
\newcommand*{\DTLidxFormatSeeItem}[1]{%
  \DTLgidxFetchEntry{\datagidx@value}{#1}{Name}%
  \datagidxlink{#1}%
  {%
    \datagidx@value
  }%
}
```

\DTLidxSeeSep Separator in cross-reference list.

```
\newcommand*{\DTLidxSeeSep}{, }
```

TLidxSeeLastSep Final separator in cross-reference list.

```
\newcommand*{\DTLidxSeeLastSep}{ \& }
```

DoSeeOrLocation You should have both a "see" list and a location list. This checks if \See is null. If it isn't null, it does the "see" part, otherwise it deals with the location list.

```
\newcommand*{\DTLgidxDoSeeOrLocation}{%
  \DTLifnull\See
  {%
```

\See is null. Do we have a location?

```
     \ifdefempty\Location
     {%
     }%
     {%
       \DTLgidxPreLocation
       \DTLgidxLocation
     }%
  }%
  {%
```

\See is not null, so do the cross-reference.

```
     \DTLgidxSee
  }%
}
```

dx@sortchildren The list of child labels needs to be sorted so that the child list follows the same ordering as the database.

```
\newcommand*{\datagidx@sortchildren}{%
  \def\datagidx@sortedlist{}%
  \@for\Label:=\Children\do
  {%
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\DTLgidxCurrentdb}%
      {\dtlcolumnindex{\DTLgidxCurrentdb}{Label}}%
      {\Label}%
    }%
    \do@getrow
```

Row index is stored in `\dtlrownum`. Is the sorted list empty?

```
  \ifdefempty\datagidx@sortedlist
  {%
```

Yes, it's empty.

```
    \edef\datagidx@newsortedlist{{\number\dtlrownum}{\Label}}%
  }%
  {%
```

No, it's not empty. Need to insert into list.

```
    \def\datagidx@newsortedlist{}%
    \@for\@datagidx@thisval:=\datagidx@sortedlist\do
    {%
```

Get the index:

```
      \edef\datagidx@thisidx{\expandafter\@firstoftwo\@datagidx@thisval}%
```

Is index greater than `\dtlrownum`?

```
      \ifnum\datagidx@thisidx>\dtlrownum\relax
```

Yes, it is. So insert here.

```
        \ifdefempty\datagidx@newsortedlist
        {%
          \eappto\datagidx@newsortedlist
          {%
            {\number\dtlrownum}{\Label},\@datagidx@thisval
          }%
        }%
        {%
          \eappto\datagidx@newsortedlist
          {%
            ,{\number\dtlrownum}{\Label},\@datagidx@thisval
          }%
        }%
```

Break out of inner loop.

```
        \@endfortrue
      \else
        \ifdefempty\datagidx@newsortedlist
        {%
          \edef\datagidx@newsortedlist{%
            \@datagidx@thisval
          }%
        }%
        {%
          \eappto\datagidx@newsortedlist
          {%
```

```
                ,\@datagidx@thisval
              }%
            }%
          \fi
        }%
```

Was the loop ended prematurely?

```
        \if@endfor
```

If loop was ended on the last iteration, `\@forremainder` will be empty and there's nothing left to do.

```
          \ifdefempty\@forremainder
          {%
          }%
          {%
```

Loop prematurely ended, so append remainder to list. newsortedlist,forremainder

```
          }%
        \else
```

Loop wasn't prematurely terminated, so new value hasn't been added. Add now.

```
          \ifdefempty\datagidx@newsortedlist
          {%
            \edef\datagidx@newsortedlist{{\number\dtlrownum}{\Label}}%
          }%
          {%
            \eappto\datagidx@newsortedlist{,{\number\dtlrownum}{\Label}}%
          }%
        \fi
      }%
```

Update.

```
      \let\datagidx@sortedlist\datagidx@newsortedlist
```

Don't break out of outer loop.

```
      \@endforfalse
    }%
  }
```

rt@foreachchild Sorted iteration through all the child labels.

```
  \newcommand{\datagidx@sort@foreachchild}[1]{%
    \datagidx@sortchildren
```

Sorted list stored in `\datagidx@sortedlist`

```
    \@for\@datagidx@thisval:=\datagidx@sortedlist\do
    {%
      \edef\Label{\expandafter\@secondoftwo\@datagidx@thisval}%
      #1%
    }%
  }
```

rt@foreachchild Unsorted iteration through all the child labels.

```
\newcommand{\datagidx@unsort@foreachchild}[1]{%
  \@for\Label:=\Children\do
  {%
    #1%
  }%
}
```

DTLgidxChildren How to display the children

```
\newcommand*{\DTLgidxChildren}{%
  \bgroup
    \DTLifnull\Children
    {}%
    {%
      \advance\datagidx@level by 1\relax
      \datagidxchildstart
      \let\Parent\Label
      \datagidx@foreachchild
      {%
        \edef\do@getrow{%
          \noexpand\dtlgetrowforvalue
          {\DTLgidxCurrentdb}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{Label}}%
          {\Label}%
        }%
        \do@getrow
        \dtlgetentryfromcurrentrow
          {\Location}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{Location}}%
        \dtlgetentryfromcurrentrow
          {\See}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{See}}%
        \dtlgetentryfromcurrentrow
          {\SeeAlso}%
          {\dtlcolumnindex{\DTLgidxCurrentdb}{SeeAlso}}%
        \DTLifnull\Location
        {%
          \DTLifnull\See
          {%
            \DTLifnull\SeeAlso
            {}%
            {%
              \datagidx@displaychild
            }%
          }%
          {%
            \datagidx@displaychild
          }%
        }%
```

```
          {%
            \datagidx@displaychild
          }%
        }%
        \datagidxchildend
      }%
    \egroup
  }
```

xgetchildfields Get the child fields from the current row.

```
\newcommand*{\datagidxgetchildfields}{%
  \dtlgetentryfromcurrentrow
    {\Name}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Name}}%
  \dtlgetentryfromcurrentrow
    {\Description}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Description}}%
  \dtlgetentryfromcurrentrow
    {\Symbol}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Symbol}}%
  \dtlgetentryfromcurrentrow
    {\Long}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Long}}%
  \dtlgetentryfromcurrentrow
    {\Short}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Short}}%
  \dtlgetentryfromcurrentrow
    {\Text}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Text}}%
  \dtlgetentryfromcurrentrow
    {\Plural}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Plural}}%
  \dtlgetentryfromcurrentrow
    {\Short}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Used}}%
  \dtlgetentryfromcurrentrow
    {\Children}%
    {\dtlcolumnindex{\DTLgidxCurrentdb}{Child}}%
}
```

dx@displaychild

```
\newcommand*{\datagidx@displaychild}{%
  \datagidxgetchildfields
  \datagidxchilditem
}
```

Define some keys for \newgloss:

atagidx@heading Indicates how to format the heading in the glossary/index.

```
\ifdef{\chapter}
{%
  \newcommand*{\datagidx@heading}{\chapter*}
}%
{%
  \newcommand*{\datagidx@heading}{\section*}
}
```

TLgidxNoHeading Allow user to suppress the heading. (So to suppress the heading do heading=\DTLgidxNoHeading).

```
\let\DTLgidxNoHeading\@gobble
```

idx@postheading Indicates what to do immediately after the heading.

```
\newcommand*{\datagidx@postheading}{}
```

agidx@multicols Should we use multicols or multicols*?

```
\newcommand*{\datagidx@multicols}{multicols}
```

\datagidx@sort Indicates how to sort the glossary/index. Defaults to word order.

```
\newcommand*{\datagidx@sort}{%
  \dtlsort{Sort,FirstId}{\DTLgidxCurrentdb}{\dtlwordindexcompare}%
}
```

\@idxitem Some classes, such as beamer, don't define \@idxitem so if it's not already defined, define it here.

```
\providecommand{\@idxitem}{\par\hangindent 40\p@}
```

\datagidxstart Indicates what to do at the start of the glossary/index.

```
\newcommand*{\datagidxstart}%
{%
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \let\item\@idxitem
}
```

\datagidxend Indicates what to do at the end of the glossary/index.

```
\newcommand*{\datagidxend}{\egroup}
```

\datagidxtarget Provide a means to add a hypertarget if \hypertarget has been defined.

```
\newcommand*{\@datagidxtarget}[2]{%
  \ifdef\hypertarget
  {%
    \bgroup
      \let\glsadd\@gobble
      \settoheight\dimen@{#2}%
      \raisebox{\dimen@}{\hypertarget{#1}{}}%
    \egroup
  }%
  {%
  }%
  #2%
}
\newcommand*{\datagidxtarget}{\@datagidxtarget}
```

\datagidxlink Provide a means to add a link if \hyperlink has been defined.

```
\newcommand*{\@datagidxlink}[2]{%
  \ifdef\hyperlink
  {%
    \hyperlink{#1}{#2}%
  }%
  {%
    #2%
  }%
}
\newcommand*{\datagidxlink}{\@datagidxlink}
```

gidxEnableHyper Enable hyperlinks (if they are defined).

```
\newcommand*{\DTLgidxEnableHyper}{%
  \let\datagidxtarget\@datagidxtarget
  \let\datagidxlink\@datagidxlink
}
```

idxDisableHyper Disable hyperlinks (if they are defined).

```
\newcommand*{\DTLgidxDisableHyper}{%
  \let\datagidxtarget\@secondoftwo
  \let\datagidxlink\@secondoftwo
}
```

atagidxgroupsep Indicates what to do between groups (after the previous group and before the header of the next group).

```
\newcommand*{\datagidxgroupsep}{}
```

gidxgroupheader Indicates what to do at the start of a group. (The current group label can be accessed via \datagidxcurrentgroup and the previous group label can be accessed via \datagidxprevgroup.)

```
\newcommand*{\datagidxgroupheader}{}
```

\datagidxitem Indicates what to do at the start of each item of the glossary/index.

```
\newcommand*{\datagidxitem}{}%
```

agidxchildstart Indicates what to do at the start of the child glossary/index.

```
\newcommand*{\datagidxchildstart}{}
```

atagidxchildend Indicates what to do at the end of the child glossary/index.

```
\newcommand*{\datagidxchildend}{}
```

tagidxchilditem Indicates what to do at the start of each item of the child glossary/index.

```
\newcommand*{\datagidxchilditem}{}%
```

idxseealsostart Indicates what to do at the start of the “see also” list.

```
\newcommand*{\datagidxseealsostart}{}
```

agidxseealsoend Indicates what to do at the end of the “see also” list.

```
\newcommand*{\datagidxseealsoend}{}
```

@doifsymlocwidth

```
\datagidx@doifsymlocwidth{⟨indent⟩}{⟨Name code⟩}{⟨Location code⟩}
```

What to do if both the symbol width and the location width have been set.

```
\newcommand*{\datagidx@doifsymlocwidth}[3]{%
```

Calculate remaining space left for the description.

```
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxsymbolwidth}%
  \addtolength{\dtl@tmplength}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \if@datagidxsymbolleft
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
  \else
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
  \fi
  \DTLgidxPreLocation
  \begin{minipage}[t]{\datagidxlocationwidth}%
    \datagidxlocalign
    \let\DTLgidxPreLocation\@empty
    #3%
  \end{minipage}%
}
```

idx@doiflocwidth

\datagidx@doiflocwidth{⟨*indent*⟩}{⟨*Name code*⟩}{⟨*Location code*⟩}

What to do if only the location width has been set.

```
\newcommand*{\datagidx@doiflocwidth}[3]{%
```

Calculate remaining space left for the symbol and description.

```
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \begin{minipage}[t]{\dtl@tmplength}%
    \DTLgidxSymbolDescription
  \end{minipage}%
  \DTLgidxPreLocation
  \begin{minipage}[t]{\datagidxlocationwidth}%
    \datagidxlocalign
    \let\DTLgidxPreLocation\@empty
    #3%
  \end{minipage}%
}
```

idx@doifsymwidth

\datagidx@doifsymwidth{⟨*indent*⟩}{⟨*Name code*⟩}{⟨*Location code*⟩}

What to do if only the location width has been set.

```
\newcommand*{\datagidx@doifsymwidth}[3]{%
```

Calculate remaining space left for the description and location.

```
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-#1}%
  \settowidth{\dimen@}{#2}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \addtolength{\dtl@tmplength}{-\datagidxsymbolwidth}%
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \if@datagidxsymbolleft
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
      #3%
    \end{minipage}%
  \else
    \begin{minipage}[t]{\dtl@tmplength}%
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescRight
    \end{minipage}%
    \DTLgidxSymDescSep
    \begin{minipage}[t]{\datagidxsymbolwidth}%
      \datagidxsymalign
      \let\DTLgidxSymDescSep\@empty
      \DTLgidxSymbolDescLeft
```

This arrangement may look a bit weird.

```
      #3%
    \end{minipage}%
  \fi
}
```

atagidxlocalign

Alignment of the location when the location width has been set.

```
\newcommand*{\datagidxlocalign}{\raggedleft}
```

atagidxsymalign

Alignment of the symbol when the symbol width has been set.

```
\newcommand*{\datagidxsymalign}{\centering}
```

5.3.1 Predefined styles

atagidxsetstyle

Sets the current index/glossary style

```
\newcommand*{\datagidxsetstyle}[1]{%
  \ifcsdef{datagidx@style@#1}%
  {%
    \csuse{datagidx@style@#1}%
  }%
  {%
    \PackageError{datagidx}{Unknown style ‘#1’}{}%
  }%
}
```

index

idx@style@index

Basic index style.

```
  \newcommand*{\datagidx@style@index}{%
    \renewcommand*{\datagidxstart}%
    {%
      \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
```

Index columns are usually too narrow for fully justified text.

```
      \raggedright
      \let\item\@idxitem
```

Have the symbol or location widths been set?

```
      \ifdim\datagidxsymbolwidth>0pt\relax
```

Symbol width has been set Has the location width been set?

```
        \ifdim\datagidxlocationwidth>0pt\relax
```

Both have been set.

```
          \def\datagidx@item@body{%
            \datagidx@doifsymlocwidth{0pt}%
             {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
             {%
               \DTLgidxDoSeeOrLocation
             }%
           }%
        \else
```

Location width hasn't been set.

```
          \def\datagidx@item@body{%
            \datagidx@doiflocwidth{0pt}%
             {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
             {%
               \DTLgidxDoSeeOrLocation
             }%
           }%
        \fi
      \else
```

Symbol width hasn't been set Has the location width been set?

```
        \ifdim\datagidxlocationwidth>0pt\relax
```

Location width has been set.

```
          \def\datagidx@item@body{%
            \datagidx@doiflocwidth{0pt}%
             {\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
             {%
               \DTLgidxDoSeeOrLocation
             }%
           }%
        \else
```

Neither have been set.

```
          \def\datagidx@item@body{%
            \DTLgidxSymbolDescription
            \DTLgidxDoSeeOrLocation
          }%
        \fi
      \fi
    }%
    \renewcommand*{\datagidxend}{\egroup}%
    \renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
    \renewcommand{\datagidxgroupheader}{%
      \ifdatagidxshowgroups
        \item
         \makebox[\linewidth]%
         {%
           \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
         }%
         \DTLpar\nobreak\@afterheading
      \fi
    }%
    \renewcommand*{\datagidxitem}{%
```

Is this the start of a new group?

```
      \ifdefempty\datagidxprevgroup
      {%
```

First item of the list.

```
        \datagidxgroupheader
      }%
      {%
```

Not the first item of the list. Is this item's group the same as the last item's group?

```
        \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
        {%
```

Same, so do nothing.

```
        }%
        {%
```

Different, so do the separator and the header.

```
          \datagidxgroupsep
          \datagidxgroupheader
        }%
      }%
```

Now get on with this item.

```
      \item
      \datagidxtarget{\Label}%
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      }%
      \DTLgidxPostName
```

```
    \datagidx@item@body
    \DTLgidxChildrenSeeAlso
  }%
  \renewcommand*{\datagidxchildstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \let\item\@idxitem
  }%
  \renewcommand*{\datagidxchildend}{\egroup}%
  \renewcommand*{\datagidxchilditem}{%
    \setlength{\dimen@}{\datagidxindent}%
    \multiply\dimen@ by \datagidx@level\relax
    \@idxitem\hspace*{\dimen@}%
    \refstepcounter{DTLgidxChildCount}%
    \datagidxtarget{\Label}%
    {%
     \DTLgidxChildStyle
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
  }%
  \renewcommand*{\datagidxseealsostart}%
  {%
    \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
      \setlength{\dimen@}{\datagidxindent}%
      \advance\datagidx@level by 1\relax
      \multiply\dimen@ by \datagidx@level\relax
      \@idxitem\hspace*{\dimen@}%
  }%
  \renewcommand{\datagidxseealsoend}{\egroup}%
}
```

Make this the default style:

```
\datagidx@style@index
```

indexalign

Similar to index style but aligns the descriptions.

tyle@indexalign

```
\newcommand*{\datagidx@style@indexalign}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \setlength{\datagidxnamewidth}{0pt}%
    \DTLforeach*{\DTLgidxCurrentdb}%
      {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
       \Parent=Parent}%
    {%
       \DTLifnull{\Parent}%
       {%
         \datagidx@doifdisplayed
         {%
           \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           \ifdim\dimen@>\datagidxnamewidth\relax
              \datagidxnamewidth=\dimen@\relax
           \fi
         }%
       }%
       {}%
    }%
    \settowidth{\dimen@}{\DTLgidxPostName}%
    \addtolength{\datagidxnamewidth}{\dimen@}%
    \setlength{\datagidxdescwidth}{\linewidth}%
    \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
    \ifdim\datagidxsymbolwidth>0pt\relax
      \addtolength{\datagidxdescwidth}{-\datagidxsymbolwidth}%
      \settowidth{\dimen@}{\DTLgidxSymDescSep}%
      \addtolength{\datagidxdescwidth}{-\dimen@}%
    \fi
    \ifdim\datagidxlocationwidth>0pt\relax
      \addtolength{\datagidxdescwidth}{-\datagidxlocationwidth}%
      \settowidth{\dimen@}{\DTLgidxPreLocation}%
      \addtolength{\datagidxdescwidth}{-\dimen@}%
    \fi
```

Has the symbol width been set?

```
    \ifdim\datagidxsymbolwidth>0pt\relax
```

Yes, symbol width has been set. Has the location width been set?

```
      \ifdim\datagidxlocationwidth>0pt\relax
```

Both symbol and location widths have been set.

```
        \if@datagidxsymbolleft
```

Symbol is on the left.

```
          \def\datagidx@item@body{%
            \begin{minipage}[t]{\datagidxsymbolwidth}%
              \datagidxsymalign
```

```
        \let\DTLgidxSymDescSep\@empty
        \DTLgidxSymbolDescLeft
      \end{minipage}%
      \DTLgidxSymDescSep
      \begin{minipage}[t]{\datagidxdescwidth}%
        \let\DTLgidxSymDescSep\@empty
        \setlength{\parskip}{0pt plus 0.3pt}%
        \DTLgidxSymbolDescRight
      \end{minipage}%
      \DTLgidxPreLocation
      \begin{minipage}[t]{\datagidxlocationwidth}%
        \datagidxlocalign
        \let\DTLgidxPreLocation\@empty
        \DTLgidxDoSeeOrLocation
      \end{minipage}%
     }%
   \else
```

Symbol is on the right.

```
    \def\datagidx@item@body{%
      \begin{minipage}[t]{\datagidxdescwidth}%
        \let\DTLgidxSymDescSep\@empty
        \DTLgidxSymbolDescLeft
      \end{minipage}%
      \DTLgidxSymDescSep
      \begin{minipage}[t]{\datagidxsymbolwidth}%
        \datagidxsymalign
        \let\DTLgidxSymDescSep\@empty
        \setlength{\parskip}{0pt plus 0.3pt}%
        \DTLgidxSymbolDescRight
      \end{minipage}%
      \DTLgidxPreLocation
      \begin{minipage}[t]{\datagidxlocationwidth}%
        \datagidxlocalign
        \let\DTLgidxPreLocation\@empty
        \DTLgidxDoSeeOrLocation
      \end{minipage}%
     }%
   \fi
\else
```

Location width hasn't been set. (Only symbol width has been set.)

```
  \if@datagidxsymbolleft
    \def\datagidx@item@body{%
      \begin{minipage}[t]{\datagidxsymbolwidth}%
        \datagidxsymalign
        \let\DTLgidxSymDescSep\@empty
        \DTLgidxSymbolDescLeft
      \end{minipage}%
      \DTLgidxSymDescSep
```

```
              \begin{minipage}[t]{\datagidxdescwidth}%
                \let\DTLgidxSymDescSep\@empty
                \setlength{\parskip}{0pt plus 0.3pt}%
                \DTLgidxSymbolDescRight
                \DTLgidxDoSeeOrLocation
              \end{minipage}%
           }%
          \else
```

Symbol is on the right. This combination may look weird.

```
            \def\datagidx@item@body{%
              \begin{minipage}[t]{\datagidxdescwidth}%
                \let\DTLgidxSymDescSep\@empty
                \DTLgidxSymbolDescLeft
              \end{minipage}%
              \DTLgidxSymDescSep
              \begin{minipage}[t]{\datagidxsymbolwidth}%
                \datagidxsymalign
                \let\DTLgidxSymDescSep\@empty
                \setlength{\parskip}{0pt plus 0.3pt}%
                \DTLgidxSymbolDescRight
                \DTLgidxDoSeeOrLocation
              \end{minipage}%
           }%
          \fi
        \fi
      \else
```

Symbol width hasn't been set. Has the location width been set?

```
        \ifdim\datagidxlocationwidth>0pt\relax
```

Only location width has been set.

```
          \def\datagidx@item@body{%
            \begin{minipage}[t]{\datagidxdescwidth}%
             \setlength{\parskip}{0pt plus 0.3pt}%
             \DTLgidxSymbolDescription
            \end{minipage}%
            \DTLgidxPreLocation
            \begin{minipage}[t]{\datagidxlocationwidth}%
            \datagidxlocalign
             \let\DTLgidxPreLocation\@empty
             \DTLgidxDoSeeOrLocation
           }%
        \else
```

Neither location nor symbol widths have been set.

```
          \def\datagidx@item@body{%
            \begin{minipage}[t]{\datagidxdescwidth}%
             \setlength{\parskip}{0pt plus 0.3pt}%
             \DTLgidxSymbolDescription
             \DTLgidxDoSeeOrLocation
```

```
        \end{minipage}%
       }%
    \fi
  \fi
}%
\renewcommand*{\datagidxend}{\egroup}%
\renewcommand*{\datagidxgroupsep}{}%
\renewcommand*{\datagidxgroupheader}{}%
\renewcommand*{\datagidxitem}{%
```

Is this the start of a new group?

```
  \ifdefempty\datagidxprevgroup
  {%
```

First item of the list.

```
    \datagidxgroupheader
  }%
  {%
```

Not the first item of the list. Is this item's group the same as the last item's group?

```
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
```

Same, so do nothing.

```
    }%
    {%
```

Different, so do the separator and the header.

```
      \datagidxgroupsep
      \datagidxgroupheader
    }%
  }%
```

Get on with this item

```
  \hangindent0pt\relax
  \parindent0pt\relax
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      \DTLgidxPostName
    }%
  }%
  \datagidx@item@body
  \par
  \DTLgidxChildrenSeeAlso
  \par
}%
\renewcommand*{\datagidxchildstart}%
{%
  \bgroup
  \setlength{\dimen@}{\datagidxindent}%
  \multiply\dimen@ by \datagidx@level\relax
  \setlength{\dtl@tmplength}{\linewidth}%
  \addtolength{\dtl@tmplength}{-\dimen@}%
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \edef\item{\noexpand\parshape=1 \the\dimen@ \the\dtl@tmplength}%
  \setlength{\datagidxnamewidth}{0pt}%
  \DTLforeach*{\DTLgidxCurrentdb}%
    {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
     \Parent=Parent}%
  {%
     \DTLifnull{\Parent}%
     {%
       \datagidx@doifdisplayed
       {%
         \settowidth{\dimen@}%
         {%
           \DTLgidxChildStyle
           {%
             \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
           }%
         }%
         \ifdim\dimen@>\datagidxnamewidth\relax
            \datagidxnamewidth=\dimen@\relax
         \fi
       }%
     }%
     {}%
  }%
  \settowidth{\dimen@}{\DTLgidxChildStyle\DTLgidxPostChildName}%
  \addtolength{\datagidxnamewidth}{\dimen@}%
  \setlength{\datagidxdescwidth}{\dtl@tmplength}%
  \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
}%
\renewcommand{\datagidxchildend}{\egroup}%
\renewcommand*{\datagidxchilditem}{%
  \item
  \refstepcounter{DTLgidxChildCount}%
  \makebox[\datagidxnamewidth][l]%
  {%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxChildStyle
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
    }%
    \begin{minipage}[t]{\datagidxdescwidth}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \DTLgidxSymbolDescription
     \DTLgidxDoSeeOrLocation
     \DTLgidxChildrenSeeAlso
    \end{minipage}%
    \par
  }%
}
```

\datagidxindent Indent used by index and indexalign styles.

```
\newlength\datagidxindent
\setlength\datagidxindent{10\p@}
```

align

tagidxnamewidth Length used by align and indexalign style name.

```
\newlength\datagidxnamewidth
```

tagidxdescwidth Length used by align and indexalign style description.

```
\newlength\datagidxdescwidth
```

idx@style@align

```
\newcommand*{\datagidx@style@align}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \setlength{\datagidxnamewidth}{0pt}%
    \DTLforeach*{\DTLgidxCurrentdb}%
      {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
       \Parent=Parent}%
    {%
       \DTLifnull{\Parent}%
       {%
         \datagidx@doifdisplayed
         {%
           \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
           \ifdim\dimen@>\datagidxnamewidth\relax
              \datagidxnamewidth=\dimen@\relax
           \fi
         }%
       }%
       {}%
    }%
    \settowidth{\dimen@}{\DTLgidxPostName}%
    \addtolength{\datagidxnamewidth}{\dimen@}%
```

```
\setlength{\datagidxdescwidth}{\linewidth}%
\addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
\ifdim\datagidxsymbolwidth>0pt\relax
  \addtolength{\datagidxdescwidth}{-\datagidxsymbolwidth}%
  \settowidth{\dimen@}{\DTLgidxSymDescSep}%
  \addtolength{\datagidxdescwidth}{-\dimen@}%
\fi
\ifdim\datagidxlocationwidth>0pt\relax
  \addtolength{\datagidxdescwidth}{-\datagidxlocationwidth}%
  \settowidth{\dimen@}{\DTLgidxPreLocation}%
  \addtolength{\datagidxdescwidth}{-\dimen@}%
\fi
```

Has the symbol width been set?

```
\ifdim\datagidxsymbolwidth>0pt\relax
```

Yes, symbol width has been set. Has the location width been set?

```
  \ifdim\datagidxlocationwidth>0pt\relax
```

Both symbol and location widths have been set.

```
    \if@datagidxsymbolleft
```

Symbol is on the left.

```
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxsymbolwidth}%
          \datagidxsymalign
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
        \DTLgidxSymDescSep
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \setlength{\parskip}{0pt plus 0.3pt}%
          \DTLgidxSymbolDescRight
        \end{minipage}%
        \DTLgidxPreLocation
        \begin{minipage}[t]{\datagidxlocationwidth}%
          \datagidxlocalign
          \let\DTLgidxPreLocation\@empty
          \DTLgidxDoSeeOrLocation
          \DTLgidxChildrenSeeAlso
        \end{minipage}%
       }%
     \else
```

Symbol is on the right.

```
      \def\datagidx@item@body{%
        \begin{minipage}[t]{\datagidxdescwidth}%
          \let\DTLgidxSymDescSep\@empty
          \DTLgidxSymbolDescLeft
        \end{minipage}%
```

```
              \DTLgidxSymDescSep
              \begin{minipage}[t]{\datagidxsymbolwidth}%
                \datagidxsymalign
                \let\DTLgidxSymDescSep\@empty
                \setlength{\parskip}{0pt plus 0.3pt}%
                \DTLgidxSymbolDescRight
              \end{minipage}%
              \DTLgidxPreLocation
              \begin{minipage}[t]{\datagidxlocationwidth}%
                \datagidxlocalign
                \let\DTLgidxPreLocation\@empty
                \DTLgidxDoSeeOrLocation
                \DTLgidxChildrenSeeAlso
              \end{minipage}%
             }%
           \fi
        \else
```

Location width hasn't been set. (Only symbol width has been set.)

```
          \if@datagidxsymbolleft
            \def\datagidx@item@body{%
              \begin{minipage}[t]{\datagidxsymbolwidth}%
                \datagidxsymalign
                \let\DTLgidxSymDescSep\@empty
                \DTLgidxSymbolDescLeft
              \end{minipage}%
              \DTLgidxSymDescSep
              \begin{minipage}[t]{\datagidxdescwidth}%
                \let\DTLgidxSymDescSep\@empty
                \setlength{\parskip}{0pt plus 0.3pt}%
                \DTLgidxSymbolDescRight
                \DTLgidxDoSeeOrLocation
                \DTLgidxChildrenSeeAlso
              \end{minipage}%
           }%
          \else
```

Symbol is on the right. This combination may look weird.

```
            \def\datagidx@item@body{%
              \begin{minipage}[t]{\datagidxdescwidth}%
                \let\DTLgidxSymDescSep\@empty
                \DTLgidxSymbolDescLeft
              \end{minipage}%
              \DTLgidxSymDescSep
              \begin{minipage}[t]{\datagidxsymbolwidth}%
                \datagidxsymalign
                \let\DTLgidxSymDescSep\@empty
                \setlength{\parskip}{0pt plus 0.3pt}%
                \DTLgidxSymbolDescRight
                \DTLgidxDoSeeOrLocation
                \DTLgidxChildrenSeeAlso
             \end{minipage}%
           }%
          \fi
        \fi
      \else
```

Symbol width hasn't been set. Has the location width been set?

```
       \ifdim\datagidxlocationwidth>0pt\relax
```

Only location width has been set.

```
         \def\datagidx@item@body{%
           \begin{minipage}[t]{\datagidxdescwidth}%
            \setlength{\parskip}{0pt plus 0.3pt}%
            \DTLgidxSymbolDescription
           \end{minipage}%
           \DTLgidxPreLocation
           \begin{minipage}[t]{\datagidxlocationwidth}%
           \datagidxlocalign
            \let\DTLgidxPreLocation\@empty
            \DTLgidxDoSeeOrLocation
            \DTLgidxChildrenSeeAlso
           \end{minipage}%
          }%
       \else
```

Neither location nor symbol widths have been set.

```
         \def\datagidx@item@body{%
           \begin{minipage}[t]{\datagidxdescwidth}%
            \setlength{\parskip}{0pt plus 0.3pt}%
            \DTLgidxSymbolDescription
            \DTLgidxDoSeeOrLocation
            \DTLgidxChildrenSeeAlso
           \end{minipage}%
          }%
       \fi
     \fi
   }%
   \renewcommand*{\datagidxend}{\egroup}%
   \renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
   \renewcommand{\datagidxgroupheader}{%
     \ifdatagidxshowgroups
       \item
        \makebox[\linewidth]%
        {%
          \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
        }%
        \DTLpar\nobreak\@afterheading
     \fi
   }%
   \renewcommand*{\datagidxitem}{%
```

Is this the start of a new group?

```
      \ifdefempty\datagidxprevgroup
      {%
```

First item of the list.

```
        \datagidxgroupheader
      }%
      {%
```

Not the first item of the list. Is this item's group the same as the last item's group?

```
        \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
        {%
```

Same, so do nothing.

```
        }%
        {%
```

Different, so do the separator and the header.

```
          \datagidxgroupsep
          \datagidxgroupheader
        }%
      }%
      \hangindent0pt\relax
      \parindent0pt\relax
      \makebox[\datagidxnamewidth][l]%
      {%
        \datagidxtarget{\Label}%
        {%
          \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
          \DTLgidxPostName
        }%
      }%
       \datagidx@item@body
      \par
    }%
    \renewcommand*{\datagidxchildstart}%
    {%
      \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
      \setlength{\datagidxnamewidth}{0pt}%
      \DTLforeach*{\DTLgidxCurrentdb}%
        {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
         \Parent=Parent}%
      {%
         \DTLifnull{\Parent}%
         {%
           \datagidx@doifdisplayed
           {%
             \settowidth{\dimen@}%
             {%
               \DTLgidxChildStyle
               {%
                 \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
               }%
             }%
             \ifdim\dimen@>\datagidxnamewidth\relax
                \datagidxnamewidth=\dimen@\relax
             \fi
           }%
         }%
         {}%
      }%
      \settowidth{\dimen@}{\DTLgidxChildStyle\DTLgidxPostChildName}%
      \addtolength{\datagidxnamewidth}{\dimen@}%
      \setlength{\datagidxdescwidth}{\linewidth}%
      \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
    }%
    \renewcommand{\datagidxchildend}{\egroup}%
    \renewcommand*{\datagidxchilditem}{%
      \hangindent0pt\relax
      \parindent0pt\relax
      \refstepcounter{DTLgidxChildCount}%
      \makebox[\datagidxnamewidth][l]%
      {%
        \datagidxtarget{\Label}%
        {%
          \DTLgidxChildStyle
          {%
            \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
            \DTLgidxPostChildName
          }%
        }%
      }%
      \begin{minipage}[t]{\datagidxdescwidth}%
       \setlength{\parskip}{0pt plus 0.3pt}%
       \DTLgidxSymbolDescription
       \DTLgidxDoSeeOrLocation
       \DTLgidxChildrenSeeAlso
      \end{minipage}%
      \par
    }%
  }
```

gloss

idx@style@gloss

```
\newcommand*{\datagidx@style@gloss}{%
  \renewcommand*{\datagidxstart}%
  {%
  \bgroup
  \setlength{\parindent}{0pt}%
  \setlength{\parskip}{0pt plus 0.3pt}%
  \setlength{\datagidxnamewidth}{0pt}%
  \DTLforeach*{\DTLgidxCurrentdb}%
    {\Name=Name,\Location=Location,\See=See,\SeeAlso=SeeAlso,%
     \Parent=Parent}%
  {%
     \DTLifnull{\Parent}%
     {%
       \datagidx@doifdisplayed
       {%
         \settowidth{\dimen@}{\DTLgidxNameFont{\DTLgidxNameCase{\Name}}}%
         \ifdim\dimen@>\datagidxnamewidth\relax
            \datagidxnamewidth=\dimen@\relax
         \fi
       }%
     }%
     {}%
  }%
  \settowidth{\dimen@}{\DTLgidxPostName}%
  \addtolength{\datagidxnamewidth}{\dimen@}%
  \setlength{\datagidxdescwidth}{\linewidth}%
  \addtolength{\datagidxdescwidth}{-\datagidxnamewidth}%
}%
\renewcommand*{\datagidxend}{\egroup}%
\renewcommand*{\datagidxgroupsep}{\ifdatagidxshowgroups\indexspace\fi}%
\renewcommand{\datagidxgroupheader}{%
  \ifdatagidxshowgroups
    \item
     \makebox[\linewidth]%
     {%
       \textbf{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
     }%
     \DTLpar\nobreak\@afterheading
  \fi
}%
\renewcommand*{\datagidxitem}{%
```

Is this the start of a new group?

```
  \ifdefempty\datagidxprevgroup
  {%
```

First item of the list.

```
    \datagidxgroupheader
  }%
  {%
```

Not the first item of the list. Is this item's group the same as the last item's group?

```
    \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
    {%
```

Same, so do nothing.

```
      }%
      {%
```

Different, so do the separator and the header.

```
        \datagidxgroupsep
        \datagidxgroupheader
      }%
    }%
    \hangindent0pt\relax
    \parindent0pt\relax
    \makebox[\datagidxnamewidth][l]%
    {%
      \datagidxtarget{\Label}%
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostName
      }%
    }%
    \begin{minipage}[t]{\datagidxdescwidth}%
     \setlength{\parskip}{0pt plus 0.3pt}%
     \@tempswatrue
     \ifdefempty{\Description}%
     {%
       \ifdefempty{\Symbol}%
       {%
         \ifdefempty{\Location}{\@tempswafalse}{}%
       }%
       {}%
     }%
     {}%
     \if@tempswa
       \DTLgidxSymbolDescription
       \DTLgidxDoSeeOrLocation

     \else
       \mbox{}%
     \fi
     \DTLgidxChildrenSeeAlso
    \end{minipage}%
    \par
  }%
  \renewcommand*{\datagidxchildstart}%
  {%
    \bgroup
    \def\datagidx@childsep{}%
    \setcounter{DTLgidxChildCount}{0}%
  }%
  \renewcommand{\datagidxchildend}{\DTLgidxPostChild\egroup}%
  \renewcommand*{\datagidxchilditem}{%
```

```
    \datagidx@childsep
    \refstepcounter{DTLgidxChildCount}%
    \datagidxtarget{\Label}%
    {%
      \DTLgidxChildStyle
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
        \DTLgidxPostChildName
      }%
    }%
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
    \let\datagidx@childsep\DTLgidxChildSep
  }%
}
```

DTLgidxChildSep Separator between child entries for gloss style.

```
\newcommand*{\DTLgidxChildSep}{ }
```

TLgidxPostChild What to put at the end of child entries for gloss style.

```
\newcommand*{\DTLgidxPostChild}{}
```

DTLgidxDictHead Group header for dict style.

```
\ifdef\chapter
{%
  \newcommand\DTLgidxDictHead{%
    \chapter{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
  }%
}%
{%
  \newcommand\DTLgidxDictHead{%
    \section{\DTLgidxGroupHeaderTitle{\datagidxcurrentgroup}}%
  }%
}
```

ategoryNameFont Font used for ‘category’ entries with ‘dict’ style.

```
\newcommand*{\DTLgidxCategoryNameFont}[1]{#1}
```

gidxCategorySep Separator used with ‘dict’ style.

```
\newcommand*{\DTLgidxCategorySep}{\space}
```

xSubCategorySep Separator used with ‘dict’ style.

```
\newcommand*{\DTLgidxSubCategorySep}{\space}
```

agidxdictindent Indent used by ‘dict’ style.

```
\newcommand*{\datagidxdictindent}{1em}
```

idxDictPostItem What to do at the end of each item in the 'dict' style.

```
\newcommand{\DTLgidxDictPostItem}{\par}
```

gidx@style@dict Dictionary style. This assumes a hierarchical structure where the top level entries have a name. The next level is used to indicate a category, such as "adjective" or "noun". If there is only one meaning this level also has a description. If there is more than one meaning, each meaning should be a child of the category entry. Only third level entries are numbered. The child key is ignored in this style. The symbol is ignored. The location and symbols widths are also ignored.

```
\newcommand*{\datagidx@style@dict}{%
  \renewcommand*{\datagidxstart}%
  {%
    \bgroup
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0pt plus 0.3pt}%
    \dimen@=\linewidth
    \advance\dimen@ by -\datagidxdictindent\relax
    \dtl@tmplength=\datagidxdictindent\relax
    \xdef\datagidxdictparshape{%
      \noexpand\parshape=2 0pt \the\linewidth\space
       \the\dtl@tmplength\space \the\dimen@\relax
    }%
    \datagidx@level=1\relax
```

Index columns are usually too narrow for fully justified text.

```
    \raggedright
  }%
  \renewcommand*{\datagidxend}{\egroup}%
  \renewcommand*{\datagidxgroupsep}{}%
  \renewcommand{\datagidxgroupheader}{%
    \ifdatagidxshowgroups
      \datagidxend
      \datagidx@postend
      \DTLgidxDictHead
      \datagidx@prestart
      \datagidxstart
    \fi
  }%
  \renewcommand*{\datagidxitem}{%
```

Is this the start of a new group?

```
    \ifdefempty\datagidxprevgroup
    {%
```

First item of the list.

```
      \datagidxgroupheader
    }%
    {%
```

Not the first item of the list. Is this item's group the same as the last item's group?

```
        \ifdefequal\datagidxcurrentgroup\datagidxprevgroup
        {%
```

Same, so do nothing.

```
        }%
        {%
```

Different, so do the separator and the header.

```
          \datagidxgroupsep
          \datagidxgroupheader
        }%
      }%
```

Now get on with this item.

```
      \datagidxdictparshape
      \datagidxtarget{\Label}%
      {%
        \DTLgidxNameFont{\DTLgidxNameCase{\Name}}%
      }%
      \DTLgidxPostName
```

Initialise category separator to do nothing.

```
      \let\datagidx@catsep\@empty
      \let\datagidx@subcatsep\@empty
      \DTLgidxSymbolDescription
```

No location list.

```
      \DTLgidxChildrenSeeAlso
      \DTLgidxDictPostItem
    }%
    \renewcommand*{\datagidxchildstart}%
    {%
      \bgroup
    }%
    \renewcommand*{\datagidxchildend}{\egroup}%
    \renewcommand*{\datagidxchilditem}{%
```

Which level are we on?

```
      \ifnum\datagidx@level=2\relax
```

Category entry

```
        \datagidx@catsep
        \let\datagidx@catsep\DTLgidxCategorySep
        \let\datagidx@subcapsep\@empty
        \datagidxtarget{\Label}%
        {%
          \DTLgidxChildStyle
          {%
            \DTLgidxCategoryNameFont{\DTLgidxNameCase{\Name}}%
            \DTLgidxPostChildName
          }%
        }%
```

```
      \setcounter{DTLgidxChildCount}{0}%
    \else
```

Sub Category entry

```
      \datagidx@subcatsep
      \let\datagidx@subcatsep\DTLgidxSubCategorySep
      \refstepcounter{DTLgidxChildCount}%
      \DTLgidxChildCountLabel
      \DTLgidxPostChildName
    \fi
    \DTLgidxSymbolDescription
    \DTLgidxDoSeeOrLocation
    \DTLgidxChildrenSeeAlso
  }%
  \renewcommand*{\datagidxseealsostart}%
  {%
    \bgroup
      \setlength{\parindent}{0pt}%
      \setlength{\parskip}{0pt plus 0.3pt}%
      \setlength{\dimen@}{\datagidxindent}%
      \advance\datagidx@level by 1\relax
      \multiply\dimen@ by \datagidx@level\relax
      \@idxitem\hspace*{\dimen@}%
  }%
  \renewcommand{\datagidxseealsoend}{\egroup}%
}
```

5.3.2 Location Lists

dofirstlocation

Only display the first location in the list.

```
\newcommand*{\dtldofirstlocation}{%
  \@for\dtl@thisloc:=\Location\do{%
    \ifdefempty\dtl@thisloc
    {}%
    {%
       \expandafter\datagidx@getlocation\dtl@thisloc
       \datagidxlink{\datagidx@current@target}%
       {%
         \datagidx@formatlocation
            \datagidx@current@format\datagidx@current@locationstring
       }%
```

Only interested in the first item, so break out of loop.

```
      \@endfortrue
    }%
  }%
}
```

@formatlocation

```
\newcommand*{\datagidx@formatlocation}[2]{%
  \ifdefempty{#1}%
  {#2}%
  {%
    \ifcsdef{#1}%
    {%
      \csuse{#1}{#2}%
    }%
    {%
     \PackageWarning{datagidx}{Unknown format ‘#1’}%
      #2%
    }%
  }%
}
```

ldolocationlist Display the location list.

```
\newcommand*{\dtldolocationlist}{%
 \DTLifnull{\Location}%
 {}%
 {%
    \def\datagidx@prev@location{-1}%
    \def\datagidx@prev@locationstring{}%
    \def\datagidx@prev@format{}%
    \def\datagidx@prev@locationformat{}%
    \def\datagidx@prev@prefix{}%
    \def\datagidx@prev@target{}%
    \def\datagidx@location@sep{}%
    \def\datagidx@location@start{-1}%
    \expandafter\forcsvlist\expandafter\datagidx@parse@location
      \expandafter{\Location}%
    \do@prevlocation % tidy up loose ends
  }%
}
```

@dtl@sequential Conditional to keep track of sequences.

```
\newif\if@dtl@sequential
```

tagidx@getlocdo Handler for \datagidx@docomplist

```
\newcommand*\datagidx@getlocdo[1]{%
  \ifdefempty\datagidx@current@location
  {}%
  {%
    \eappto\datagidx@current@prefix{%
      \datagidx@current@location\datagidx@compositor
    }%
  }%
  \def\datagidx@current@location{#1}%
}
```

idx@getlocation Get the location and store in `\current@location`:

```
\def\datagidx@getlocation[#1]#2#3{%
```

Store the original value.

```
  \def\datagidx@current@locationstring{#2}%
```

```
  \bgroup
    \datagidx@escapelocationformat
    \xdef\datagidx@current@locationformat{#2}%
    \datagidx@clearlocationformat
    \xdef\datagidx@current@location{#2}%
  \egroup
```

If the location contains a compositor, we need to get the final element and store the rest as a prefix:

```
  \let\datagidx@list\datagidx@current@location
  \def\datagidx@current@prefix{}%
  \def\datagidx@current@location{}%
  \let\do\datagidx@getlocdo
  \expandafter\datagidx@docomplist
   \expandafter{\datagidx@list}%
```

Store the format:

```
  \def\datagidx@current@format{#1}%
```

Store the target:

```
  \def\datagidx@current@target{#3}%
}
```

@parse@location Parses the location list (given in the argument).

```
\newcommand*{\datagidx@parse@location}[1]{%
```

Parse location format.

```
  \datagidx@getlocation#1\relax
```

If this is the same as the previous location, do nothing.

```
  \ifdefequal{\datagidx@prev@locationstring}{\datagidx@current@locationstring}%
  {%
```

If the format is different, let the non-empty format over-ride the empty format.

```
    \ifdefequal{\datagidx@prev@format}{\datagidx@current@format}%
    {%
    }%
    {%
      \ifdefempty{\datagidx@current@format}%
      {%
```

Current format is empty, so keep previous unchanged.

```
      }%
      {%
        \ifdefempty{\datagidx@prev@format}%
        {%
```

Previous format is empty, so update.

```
           \let\datagidx@prev@format\datagidx@current@format
        }%
        {%
           \PackageWarning{datagidx}%
           {%
              Conflicting location formats ‘\datagidx@prev@format’ and
              ‘\datagidx@current@format’ for location ‘\datagidx@current@location’%
           }%
        }%
      }%
    }%
  }%
  {%
    \@datagidx@parse@location
  }%
}
```

@parse@location

```
\newcommand*{\@datagidx@parse@location}{%
```

Check if we have a sequence.

```
  \@dtl@sequentialtrue
```

A change in font format breaks the sequence.

```
  \ifdefequal{\datagidx@prev@format}{\datagidx@current@format}%
  {%
```

A change in location format breaks the sequence.

```
    \ifdefequal{\datagidx@prev@locationformat}{\datagidx@current@locationformat}%
    {%
```

A change in prefix breaks the sequence.

```
      \ifdefequal{\datagidx@prev@prefix}{\datagidx@current@prefix}%
      {%
      }%
      {%
```

Prefixes are different, so not a sequence.

```
        \@dtl@sequentialfalse
      }%
    }%
    {%
```

Formats are different, so not a sequence.

```
      \@dtl@sequentialfalse
    }%
  }%
  {%
```

Formats are different, so not a sequence.

```
    \@dtl@sequentialfalse
```

```
}%
\if@dtl@sequential
```

Is this location one more than the previous location?

```
  \ifnumequal{\datagidx@prev@location+1}{\datagidx@current@location}%
  {%
```

It is one more than previous value. Is this location the same type as the previous location?

```
     \ifdefequal
       \datagidx@current@locationformat
       \datagidx@prev@locationformat
    {%
```

They are the same, so we have a sequence.

```
      \@dtl@sequentialtrue
    }%
    {%
```

They aren't the same, so we don't have a sequence.

```
           \@dtl@sequentialfalse
    }%
  }%
  {%
     \@dtl@sequentialfalse
  }%
\fi
```

Has the sequence flag been set?

```
\if@dtl@sequential
```

Yes, we have a sequence. Has the start of the sequence been set?

```
   \ifnumequal{\datagidx@location@start}{-1}%
   {%
```

No it hasn't, so set it

```
     \let\datagidx@location@start\datagidx@prev@location
     \let\datagidx@location@startval\datagidx@prev@locationstring
     \let\datagidx@location@format\datagidx@prev@format
     \let\datagidx@location@target\datagidx@prev@target
   }%
   {%
   }%
\else
```

We don't have a sequence, so do the previous location.

```
  \do@prevlocation
\fi
```

Update previous location macros to this location.

```
\let\datagidx@prev@location\datagidx@current@location
\let\datagidx@prev@format\datagidx@current@format
\let\datagidx@prev@prefix\datagidx@current@prefix
\let\datagidx@prev@locationformat\datagidx@current@locationformat
```

```
  \let\datagidx@prev@locationstring\datagidx@current@locationstring
  \let\datagidx@prev@target\datagidx@current@target
}
```

gidxLocationSep Separator between locations.

```
\newcommand*{\DTLgidxLocationSep}{, }
```

TLgidxLocationF How to format a location list consisting of only two locations.

```
\newcommand*{\DTLgidxLocationF}[2]{%
  #1\DTLgidxLocationSep#2%
}
```

LgidxLocationFF How to format a location list consisting of three or more locations.

```
\newcommand*{\DTLgidxLocationFF}[2]{%
  #1--#2%
}
```

do@prevlocation Do the previous location in the current list.

```
\newcommand*{\do@prevlocation}{%
```

Have we come to the end of a sequence?

```
  \ifnumequal{\datagidx@location@start}{-1}%
  {%
```

Not the end of a sequence.

```
     \ifdefempty{\datagidx@prev@locationstring}%
     {}%
     {%
       \datagidx@location@sep
       \datagidxlink{\datagidx@prev@target}%
       {%
         \datagidx@formatlocation
           \datagidx@prev@format\datagidx@prev@locationstring
       }%
       \def\datagidx@location@sep{\DTLgidxLocationSep}%
     }%
  }%
  {%
```

At the end of a sequence.

```
    \datagidx@location@sep
    \do@locrange
    \def\datagidx@location@sep{\DTLgidxLocationSep}%
    \def\datagidx@location@start{-1}%
  }%
}
```

\do@locrange Format the location range.

```
\newcommand*{\do@locrange}{%
```

Are the start and end locations 2 or more apart?

```
  \ifnumgreater{\datagidx@prev@location}{\datagidx@location@start+1}%
  {%
```

Yes, they are, so form a range:

```
     \DTLgidxLocationFF
       {%
         \datagidxlink{\datagidx@location@target}%
         {%
           \datagidx@formatlocation
             \datagidx@location@format\datagidx@location@startval
         }%
       }%
       {%
         \datagidxlink{\datagidx@prev@target}%
         {%
           \datagidx@formatlocation
             \datagidx@prev@format\datagidx@prev@locationstring
         }%
       }%
  }%
  {%
```

No, they aren't so don't form a range:

```
     \DTLgidxLocationF
       {%
         \datagidxlink{\datagidx@location@target}%
         {%
           \datagidx@formatlocation
             \datagidx@location@format\datagidx@location@startval
         }%
       }%
       {%
         \datagidxlink{\datagidx@prev@target}%
         {%
           \datagidx@formatlocation
             \datagidx@prev@format\datagidx@prev@locationstring
         }%
       }%
  }%
}
```

5.4 Defining New Glossary/Index Databases

defaultdatabase The default database to which terms should be added.

```
\newcommand*{\datagidx@defaultdatabase}{}
```

idxSetDefaultDB Allow user to set the default database

```
\newcommand*{\DTLgidxSetDefaultDB}[1]{%
```

```
  \renewcommand*{\datagidx@defaultdatabase}{#1}%
}
```

Define keys for \newgidx:

```
\define@key{newgloss}{heading}{\renewcommand*{\datagidx@heading}{#1}}
\define@key{newgloss}{postheading}{%
  \renewcommand*{\datagidx@postheading}{#1}%
}
```

datagidxbalance

```
\newif\ifdatagidxbalance
\datagidxbalancetrue

\define@choicekey{newgloss}{balance}[\val\nr]{true,false}[true]{%
 \ifcase\nr\relax
   \renewcommand*{\datagidx@multicols}{multicols}%
   \datagidxbalancetrue
 \or
   \renewcommand*{\datagidx@multicols}{multicols*}%
   \datagidxbalancefalse
 \fi
}
\define@key{newgloss}{sort}{\renewcommand*{\datagidx@sort}{#1}}
```

Default style is ‘index’:

```
\newcommand*{\datagidx@style}{index}
\define@key{newgloss}{style}{\renewcommand*{\datagidx@style}{#1}}
```

Define conditional to determine whether or not to show group headers and do sep. (Default is false.)

agidxshowgroups

```
\newif\ifdatagidxshowgroups
\newcommand*{\datagidx@showgroups}{false}

\define@choicekey{newgloss}{showgroups}{true,false}[true]%
{%
  \renewcommand{\datagidx@showgroups}{#1}%
}%
```

\newgidx

\newgloss[⟨*options*⟩]{⟨*database name*⟩}{⟨*title*⟩}

Define \newgidx if it hasn’t already been defined by the ‘highopt’ optimize setting.

```
\ifundef\newgidx
{%
  \newcommand*{\newgidx}{\datagidx@newgidx}
}%
{}
```

May only be used in the preamble (otherwise the entries will be undefined when their locations are read from the aux file).

```
\@onlypreamble\newgidx
```

highopt@newgidx The behaviour of `\newgidx` when the ‘highopt’ optimize option has been set.

```
\newcommand*{\datagidx@highopt@newgidx}[3][]{%
```

Get the file name:

```
  \edef\datagidx@indexfilename{\datagidxhighoptfilename{#2}}%
```

Has the file been created?

```
  \IfFileExists{\datagidx@indexfilename}%
  {%
```

File does exists. Load it.

```
    \input{\datagidx@indexfilename}%
```

Update the ‘datagidx’ database.

```
    \bgroup
      \setkeys{newgloss}{#1}%
      \datagidx@newgidx@update{#2}{#3}%
    \egroup
  }%
  {%
```

File doesn’t exist. Behave as normal.

```
    \datagidx@newgidx[#1]{#2}{#3}%
  }%
}
```

\loadgidx

```
\loadgidx[⟨options⟩]{⟨filename⟩}{⟨title⟩}
```

Loads a datagidx database.

```
\newcommand*{\loadgidx}[3][]{%
```

Load database:

```
  \input{#2}%
```

Update the ‘datagidx’ database. (Assume database is already sorted.)

```
  \bgroup
    \setkeys{newgloss}{sort={},#1}%
    \expandafter\datagidx@newgidx@update\expandafter
      {\dtllastloadeddb}{#3}%
  \egroup
```

Set this as the default database:

```
  \edef\datagidx@defaultdatabase{\dtllastloadeddb}%
```

Assign labels to this database.

```
    \dtlforcolumn{\Label}{\dtllastloadeddb}{Label}%
    {%
      \csxdef{datagidxentry@\Label}{\dtllastloadeddb}%
    }%
  }
```

May only be used in the preamble (otherwise the entries will be undefined when their locations are read from the aux file).

```
  \@onlypreamble\loadgidx
```

atagidx@newgidx The normal behaviour of \newgidx

```
  \newcommand*{\datagidx@newgidx}[3][]{%
  \bgroup
    \setkeys{newgloss}{#1}%
```

If no default database has been identified, set the default to this database.

```
    \ifdefempty{\datagidx@defaultdatabase}%
    {\xdef\datagidx@defaultdatabase{#2}}%
    {}%
    \DTLgnewdb{#2}%
    \DTLaddcolumn{#2}{Label}%
    \DTLaddcolumn{#2}{Location}%
    \DTLaddcolumn{#2}{CurrentLocation}%
    \DTLaddcolumn{#2}{FirstId}%
    \DTLaddcolumn{#2}{Name}%
    \DTLaddcolumn{#2}{Text}%
    \DTLaddcolumn{#2}{Parent}%
    \DTLaddcolumn{#2}{Child}%
    \DTLaddcolumn{#2}{Description}%
    \DTLaddcolumn{#2}{Used}%
    \DTLaddcolumn{#2}{Symbol}%
    \DTLaddcolumn{#2}{Long}%
    \DTLaddcolumn{#2}{Short}%
    \DTLaddcolumn{#2}{See}%
    \DTLaddcolumn{#2}{SeeAlso}%
    \datagidx@newgidx@update{#2}{#3}%
  \egroup
  }
```

@newgidx@update Update the 'datagidx' database.

```
  \newcommand*{\datagidx@newgidx@update}[2]{%
    \DTLnewrow{datagidx}%
    \DTLnewdbentry{datagidx}{Glossary}{#1}%
    \DTLnewdbentry{datagidx}{Title}{#2}%
    {%
      \dtlexpandnewvalue
      \DTLnewdbentry{datagidx}{Heading}{\expandonce\datagidx@heading}%
      \DTLnewdbentry{datagidx}{PostHeading}{\expandonce\datagidx@postheading}%
      \DTLnewdbentry{datagidx}{MultiCols}{\expandonce\datagidx@multicols}%
      \DTLnewdbentry{datagidx}{Sort}{\expandonce\datagidx@sort}%
      \DTLnewdbentry{datagidx}{Style}{\expandonce\datagidx@style}%
      \DTLnewdbentry{datagidx}{ShowGroups}{\expandonce\datagidx@showgroups}%
    }%
  }
```

5.5 Defining New Terms

5.5.1 Options

Define some keys for \newterm:

\newterm@label

```
\newcommand*{\newterm@label}{}
\define@key{newterm}{label}{\renewcommand*{\newterm@label}{#1}}
```

\newterm@parent

```
\newcommand*{\newterm@parent}{}
\define@key{newterm}{parent}{\renewcommand*{\newterm@parent}{#1}}
```

\newterm@text

```
\newcommand*{\newterm@text}{}
\define@key{newterm}{text}{\renewcommand*{\newterm@text}{#1}}
```

erm@description

```
\newcommand*{\newterm@description}{}
\define@key{newterm}{description}{%
  \renewcommand*{\newterm@description}{#1}%
}
```

\newterm@plural

```
\define@key{newterm}{plural}{\def\newterm@plural{#1}}
```

\newterm@sort

```
\newcommand*{\newterm@sort}{}
\define@key{newterm}{sort}{\renewcommand*{\newterm@sort}{#1}}
```

\newterm@symbol

```
\newcommand*{\newterm@symbol}{}
\define@key{newterm}{symbol}{\renewcommand*{\newterm@symbol}{#1}}
```

ewterm@database

```
\newcommand*{\newterm@database}{}
\define@key{newterm}{database}{\renewcommand*{\newterm@database}{#1}}
```

\newterm@long

```
\newcommand*{\newterm@long}{}
\define@key{newterm}{long}{%
  \renewcommand*{\newterm@long}{#1}%
  \def\newterm@longplural{#1s}%
}
```

\newterm@short

```
\newcommand*{\newterm@short}{}
\define@key{newterm}{short}{%
  \renewcommand*{\newterm@short}{#1}%
  \def\newterm@shortplural{#1s}%
}
```

term@longplural

```
\define@key{newterm}{longplural}{%
  \def\newterm@longplural{#1}%
}
```

erm@shortplural

```
\define@key{newterm}{shortplural}{%
  \def\newterm@shortplural{#1}%
}
```

\newterm@see “see” should not be used with a location list. If you have a location list and want a cross-reference use “see also” instead.

```
\newcommand*{\newterm@see}{}
\define@key{newterm}{see}{%
  \renewcommand*{\newterm@see}{#1}%
}
```

newterm@seealso “see also” should be used with a location list (or with child entries with location lists). If an entry has no location list and not child entries use “see” instead.

```
\newcommand*{\newterm@seealso}{}
\define@key{newterm}{seealso}{%
  \renewcommand*{\newterm@seealso}{#1}%
}
```

5.5.2 New Terms

rm@defaultshook

```
\newcommand*{\newterm@defaultshook}{}
```

erm@extrafields

```
\newcommand*{\newterm@extrafields}{}
```

LgidxAssignList Assignment list used by \printterms

```
\newcommand*{\DTLgidxAssignList}{%
  \Name=Name,\Description=Description,\Used=Used,\Symbol=Symbol,%
  \Long=Long,\Short=Short,\LongPlural=LongPlural,\ShortPlural=ShortPlural,%
  \Location=Location,\See=See,\SeeAlso=SeeAlso,%
  \Text=Text,\Plural=Plural,\CurrentLocation=CurrentLocation,%
  \Label=Label,\Parent=Parent,\Children=Child,\FirstId=FirstId,\Sort=Sort%
}
```

atagidxtermkeys Keys defined for \newterm corresponding to fields (“name” is added for convenience).

```
\newcommand*{\datagidxtermkeys}{%
  name,description,symbol,long,short,see,seealso,text,plural,%
  label,parent,sort%
}
```

Access keys corresponding to given fields

```
\newcommand*\@datagidx@fieldkey@Name{name}%
\newcommand*\@datagidx@fieldkey@Description{description}%
\newcommand*\@datagidx@fieldkey@Symbol{symbol}%
\newcommand*\@datagidx@fieldkey@Long{long}%
\newcommand*\@datagidx@fieldkey@Short{short}%
\newcommand*\@datagidx@fieldkey@See{see}%
\newcommand*\@datagidx@fieldkey@SeeAlso{seealso}%
\newcommand*\@datagidx@fieldkey@Text{text}%
\newcommand*\@datagidx@fieldkey@Plural{plural}%
\newcommand*\@datagidx@fieldkey@Label{label}%
\newcommand*\@datagidx@fieldkey@Parent{parent}%
\newcommand*\@datagidx@fieldkey@Sort{sort}%
```

\newtermaddfield

```
\newtermaddfield[⟨db list⟩]{⟨column key⟩}{⟨new term key⟩}{⟨default value⟩}
```

The default value may contain \field{⟨*key*⟩} to get the value of another field.

```
\newcommand*{\newtermaddfield}[4][]{%
```

If optional argument not specified, iterate over all defined glossaries/indices

```
  \ifstrempty{#1}%
  {%
    \dtlforcolumn{\datagidx@thisidx}{datagidx}{Glossary}%
    {%
      \DTLaddcolumn{\datagidx@thisidx}{#2}%
    }%
  }%
  {%
    \@for\datagidx@thisidx:=#1\do
    {%
```

```
        \DTLaddcolumn{\datagidx@thisidx}{#2}%
      }%
    }%
    \expandafter\gdef\csname newterm@#3\endcsname{}%
    \define@key{newterm}{#3}%
    {%
      \expandafter\def\csname newterm@#3\endcsname{##1}%
    }%
    \gappto\newterm@defaultshook
    {%
      \expandafter\protected@edef\csname newterm@#3\endcsname{#4}%
    }%
    \gappto\newterm@extrafields
    {%
       \protected@edef\datagidx@value{\csname newterm@#3\endcsname}%
       \DTLnewdbentry{\newterm@database}{#2}{\expandonce\datagidx@value}%
    }%
    \xappto\DTLgidxAssignList
    {%
      ,\expandafter\noexpand\csname#2\endcsname=#2%
    }%
    \xappto\datagidxtermkeys{,#3}%
    \expandafter\xdef\csname @datagidx@fieldkey@#2\endcsname{#3}%
    \xappto\datagidxgetchildfields
    {%
      \noexpand\dtlgetentryfromcurrentrow
       {\expandafter\noexpand\csname#2\endcsname}%
       {\noexpand\dtlcolumnindex{\noexpand\DTLgidxCurrentdb}{#2}}%
    }%
  }
```

ewtermlabelhook

```
  \newcommand*{\newtermlabelhook}{}
```

DTLgidxNoFormat

```
  \newcommand*{\DTLgidxNoFormat}[1]{#1}
```

\DTLgidxGobble

```
  \newcommand*{\DTLgidxGobble}[1]{}
```

xStripBackslash Argument must be a control sequence. This is stringified and the first character (The backslash) is removed).

```
  \newcommand*{\DTLgidxStripBackslash}[1]{%
    \expandafter\@gobble\string#1%
  }
```

\DTLgidxName

\DTLgidxName{⟨*forenames*⟩}{⟨*surname*⟩}

How to format a person's name in the text.

```
\newcommand*{\DTLgidxName}[2]{%
  #1\space #2%
}
```

\DTLgidxNameNum

```
\DTLgidxNameNum{⟨n⟩}
```

The argument ⟨*n*⟩ should be a number applied to a name (e.g. `James␣\DTLgidxNameNum1`). This is converted to a two-digit number for sorting but a Roman numeral for the label and in the text.

```
\newcommand*{\DTLgidxNameNum}[1]{\@Roman{#1}}
```

atagidx@namenum Conversion for sort key.

```
\newcommand*{\datagidx@namenum}[1]{\two@digits{#1}}
```

\DTLgidxPlace

```
\DTLgidxPlace{⟨country⟩}{⟨town/city⟩}
```

How to format a place in the text.

```
\newcommand*{\DTLgidxPlace}[2]{%
  #2%
}
```

\DTLgidxSubject

```
\DTLgidxSubject{⟨main⟩}{⟨category⟩}
```

How to format a subject in the text. Ignore the main part in the text.

```
\newcommand*{\DTLgidxSubject}[2]{%
  #2%
}
```

\DTLgidxOffice

```
\DTLgidxOffice{⟨office⟩}{⟨name⟩}
```

Put the office in parentheses in the document text.

```
\newcommand*{\DTLgidxOffice}[2]{%
  #2 (#1)%
}
```

\DTLgidxIgnore Show argument in document text, but disregard in the sort and label.

```
\newcommand*{\DTLgidxIgnore}[1]{#1}
```

\DTLgidxMac

```
\DTLgidxMac{⟨text⟩}
```

In the document, just does ⟨*text*⟩, but gets converted to “Mac” in the sort key. (Unless overridden by the user.)

```
\newcommand*{\DTLgidxMac}[1]{#1}
```

\datagidx@mac \DTLgidxMac gets temporarily redefined to \datagidx@mac when construction the sort key.

```
\newcommand*{\datagidx@mac}[1]{Mac}
```

\DTLgidxSaint

```
\DTLgidxSaint{⟨text⟩}
```

In the document, just does ⟨*text*⟩, but gets converted to “Saint” in the sort key. (Unless overridden by the user.)

```
\newcommand*{\DTLgidxSaint}[1]{#1}
```

\datagidx@saint \DTLgidxMac gets temporarily redefined to \datagidx@saint when construction the sort key.

```
\newcommand*{\datagidx@saint}[1]{Saint}
```

\DTLgidxRank

```
\DTLgidxRank{⟨rank⟩}{⟨forenames⟩}
```

A person’s title, rank or sanctity should be ignored when sorting.

```
\newcommand*{\DTLgidxRank}[2]{#1~#2}
```

\datagidx@rank \DTLidxRank gets temporarily redefined to \datagidx@rank when constructing the sort key. An extra dot is added to the end to ensure names without a rank are sorted before identical names with a rank.

```
\newcommand*{\datagidx@rank}[2]{#2.}
```

\DTLgidxParticle

```
\DTLgidxParticle{⟨particle⟩}{⟨surname⟩}
```

A particle such as “of”, “de” or “von” should be ignored when sorting.

```
\newcommand*{\DTLgidxParticle}[2]{#1~#2}
```

tagidx@particle \DTLidxParticle gets temporarily redefined to \datagidx@particle when constructing the sort key. An extra dot is added to the end to ensure names without a particle are sorted before identical names with a particle.

```
\newcommand*{\datagidx@particle}[2]{#2.}
```

agidx@bothoftwo

```
\newcommand*{\datagidx@bothoftwo}[2]{#1#2}
```

datagidx@person Used when constructing the sort key for a name.

```
\newcommand*{\datagidx@person}[2]{#2\noexpand\datatoolpersoncomma #1}
```

\datagidx@place Used when constructing the sort key for a place.

```
\newcommand*{\datagidx@place}[2]{#2\noexpand\datatoolplacecomma #1}
```

atagidx@subject Used when constructing the sort key for a place.

```
\newcommand*{\datagidx@subject}[2]{#2\noexpand\datatoolsubjectcomma #1}
```

\datagidx@paren Used when constructing the sort key for a parenthesis.

```
\newcommand*{\datagidx@paren}[1]{\noexpand\datatoolparenstart #1}
```

datagidx@invert

```
\newcommand*{\datagidx@invert}[2]{#2, #1}
```

\DTLgidxParen Parenthetical material.

```
\newcommand*{\DTLgidxParen}[1]{\space(#1)}
```

idxwordifygreek Convert commands like `\alpha` into words for indexing and labelling purposes.

```
\newcommand*{\datagidxwordifygreek}{%
  \def\alpha{alpha}%
  \def\beta{beta}%
  \def\gamma{gamma}%
  \def\delta{delta}%
  \def\epsilon{epsilon}%
  \def\varepsilon{epsilon}%
  \def\zeta{zeta}%
  \def\eta{eta}%
  \def\theta{theta}%
  \def\vartheta{theta}%
  \def\iota{iota}%
  \def\kappa{kappa}%
  \def\lambda{lambda}%
  \def\mu{mu}%
  \def\nu{nu}%
  \def\xi{xi}%
  \def\pi{pi}%
  \def\varpi{pi}%
  \def\rho{rho}%
  \def\varrho{rho}%
  \def\sigma{sigma}%
  \def\varsigma{sigma}%
  \def\tau{tau}%
  \def\upsilon{upsilon}%
  \def\phi{phi}%
  \def\varphi{phi}%
  \def\chi{chi}%
  \def\psi{psi}%
  \def\omega{omega}%
  \def\Gamma{Gamma}%
  \def\Delta{Delta}%
  \def\Theta{Theta}%
  \def\Lambda{Lambda}%
  \def\Xi{Xi}%
  \def\Pi{Pi}%
  \def\Sigma{Sigma}%
  \def\Upsilon{Upsilon}%
  \def\Phi{Phi}%
  \def\Psi{Psi}%
  \def\Omega{Omega}%
}
```

extendedtoascii Convert commands like \aa into the closest ASCII equivalent.

```
\newcommand{\datagidxextendedtoascii}{%
  \def\AE{AE}%
  \def\ae{ae}%
  \def\OE{OE}%
  \def\oe{oe}%
  \def\AA{AA}%
  \def\aa{aa}%
  \def\L{L}%
  \def\l{l}%
  \def\O{O}%
  \def\o{o}%
  \def\SS{SS}%
  \def\ss{ss}%
  \def\th{th}%
  \def\TH{TH}%
  \def\dh{dh}%
  \def\DH{DH}%
}
```

idxconvertchars Convert commands like \&.

```
\newcommand*{\datagidxconvertchars}{%
  \let~\space
  \ifdef\andname
  {%
    \let\&\andname
  }%
  {%
    \def\&{\expandafter\@gobble\string\&}%
  }%
  \def\_{\string_}%
  \def\${\string$}%
```

```
    \def\#{\expandafter\@gobble\string\#}%
    \def\%{\expandafter\@gobble\string\%}%
    \def\{{\expandafter\@gobble\string\{}%
    \def\}{\expandafter\@gobble\string\}}%
  }
```

idxstripaccents Strip accents so they don't interfere with the label and sort. If you want to write your own comparison handler macro, you'll need to redefine this if you want accented letters to be sorted differently from the unaccented version. This command should only be used within a scope. Need to test kernel version. The new 2019/10/01 version has a different definition of \UTFviii@two@octets which won't expand UTF-8 characters to \IeC in the required context. (This is really useful if you want extended characters in labels but not if they need to be stripped.) If you don't want accents stripped then redefine \datagidxstripaccents to do nothing.

```
  \@ifl@t@r\fmtversion{2019/10/01}
  {%
    \newcommand*{\datagidxstripaccents}{%
```

These redefinitions will only work with \edef or \xdef:

```
     \let\add@accent@\@secondoftwo
     \let\@text@composite@x\@secondoftwo
     \let\@tabacckludge\@secondoftwo
```

The following are need with \protected@edef or \protected@xdef:

```
     \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
     \def\IeC##1{\@gobbletwo##1}%
     \let\UTFviii@two@octets\UTFviii@two@octets@combine
    }%
  }
  {%
```

Older kernels:

```
    \newcommand*{\datagidxstripaccents}{%
     \let\add@accent@\@secondoftwo
     \let\@text@composite@x\@secondoftwo
     \let\@tabacckludge\@secondoftwo
     \expandafter\def\csname \encodingdefault-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname OT1-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname T1-cmd\endcsname##1##2##3{##3}%
     \expandafter\def\csname PD1-cmd\endcsname##1##2##3{##3}%
     \def\IeC##1{\@gobbletwo##1}%
    }%
  }
```

\newterm

\newterm[⟨*options*⟩]name

Defaults to normal behaviour.

```
\providecommand{\newterm}{\datagidx@newterm}
```

May only be used in the preamble. (Terms must be defined before the aux file is read.)

```
\@onlypreamble\newterm
```

@setfieldvalues Sets the values for all the field.

```
\newcommand{\datagidx@setfieldvalues}[2]{%
```

Set defaults.

```
  \def\newterm@name{#2}%
  \renewcommand*\newterm@label{#2}%
  \renewcommand*\newterm@text{#2}%
  \undef\newterm@plural
  \renewcommand*{\newterm@description}{}%
  \renewcommand*{\newterm@sort}{#2}%
  \renewcommand*{\newterm@symbol}{}%
  \let\newterm@database\datagidx@defaultdatabase
  \renewcommand*{\newterm@short}{#2}%
  \undef\newterm@shortplural
  \renewcommand*{\newterm@long}{#2}%
  \undef\newterm@longplural
  \renewcommand*{\newterm@see}{}%
  \renewcommand*{\newterm@seealso}{}%
  \renewcommand*{\newterm@parent}{}%
```

Allow hook to access other fields.

```
  \let\datagidx@orgfield\field
  \def\field##1{\expandafter\noexpand\csname newterm@##1\endcsname}%
```

Hook to make it easier to add extra fields.

```
  \newterm@defaultshook
  \let\field\datagidx@orgfield
```

Assign values given in optional argument.

```
  \setkeys{newterm}{#1}%
```

Temporary redefine commands likely to be contained in the name that may interfere with the label and sort.

```
  \bgroup
```

Allow users to, say, specify the name as ⟨*name*⟩\glsadd{⟨*other label*⟩} without having to specify a separate label.

```
    \let\glsadd\@gobble
```

Strip common formatting commands.

```
    \let\MakeUppercase\DTLgidxNoFormat
    \let\MakeTextUppercase\DTLgidxNoFormat
    \let\MakeLowercase\DTLgidxNoFormat
    \let\MakeTextLowercase\DTLgidxNoFormat
    \let\acronymfont\DTLgidxNoFormat
```

```
\let\textrm\DTLgidxNoFormat
\let\texttt\DTLgidxNoFormat
\let\textsf\DTLgidxNoFormat
\let\textsc\DTLgidxNoFormat
\let\textbf\DTLgidxNoFormat
\let\textmd\DTLgidxNoFormat
\let\textit\DTLgidxNoFormat
\let\textsl\DTLgidxNoFormat
\let\emph\DTLgidxNoFormat
\let\textsuperscript\DTLgidxNoFormat
```

Convert character commands like `\&`.

```
\datagidxconvertchars
```

Strip `\ensuremath`.

```
\let\ensuremath\DTLgidxNoFormat
```

Ensure that inversions are dealt with for the label.

```
\let\DTLgidxParen\@gobble
\let\DTLgidxName\@secondoftwo
\let\DTLgidxPlace\datagidx@invert
\let\DTLgidxSubject\datagidx@invert
\let\DTLgidxOffice\@secondoftwo
\let\DTLgidxParticle\datagidx@bothoftwo
```

Convert Greek maths (such as `\alpha`) to text.

```
\datagidxwordifygreek
```

Strip accent commands so they don't interfere with the label.

```
\datagidxstripaccents
\datagidxextendedtoascii
```

Allow user to hook into this.

```
\newtermlabelhook
```

There shouldn't really be any fragile commands in #2 but using `\protected@xdef` to be on the safe side.

```
\protected@xdef\newterm@label{\newterm@label}%
```

These commands behave differently for the sort key:

```
\let\DTLgidxName\datagidx@person
\let\DTLgidxPlace\datagidx@place
\let\DTLgidxSubject\datagidx@subject
\let\DTLgidxOffice\datagidx@person
\let\DTLgidxParen\datagidx@paren
\let\DTLgidxMac\datagidx@mac
\let\DTLgidxSaint\datagidx@saint
\let\DTLgidxIgnore\@gobble
\let\DTLgidxRank\datagidx@rank
\let\DTLgidxParticle\datagidx@particle
\let\DTLgidxNameNum\datagidx@namenum
```

Need to use \protected@xdef here, which means limited support for stripping accents.

```
      \protected@xdef\newterm@sort{\newterm@sort}%
   \egroup
  }
```

tagidx@add@term Add term (once all fields have been set. Argument is the name field.

```
  \newcommand*{\datagidx@add@term}[1]{%
    \global\cslet{datagidxentry@\newterm@label}{\newterm@database}%
    \DTLnewrow{\newterm@database}%
    \DTLnewdbentry{\newterm@database}{Name}{#1}%
    \DTLnewdbentry{\newterm@database}{Used}{0}%
    {%
      \dtlexpandnewvalue
      \DTLnewdbentry{\newterm@database}{Text}{\expandonce\newterm@text}%
      \DTLnewdbentry{\newterm@database}{Description}{\expandonce\newterm@description}%
      \DTLnewdbentry{\newterm@database}{Label}{\expandonce\newterm@label}%
      \DTLnewdbentry{\newterm@database}{Sort}{\expandonce\newterm@sort}%
      \DTLnewdbentry{\newterm@database}{Symbol}{\expandonce\newterm@symbol}%
      \DTLnewdbentry{\newterm@database}{Short}{\expandonce\newterm@short}%
      \DTLnewdbentry{\newterm@database}{Long}{\expandonce\newterm@long}%
      \ifundef\newterm@plural
      {%
        \DTLnewdbentry{\newterm@database}{Plural}{\expandonce\newterm@text s}%
      }%
      {%
        \DTLnewdbentry{\newterm@database}{Plural}{\expandonce\newterm@plural}%
      }%
      \ifundef\newterm@shortplural
      {%
        \DTLnewdbentry{\newterm@database}{ShortPlural}{\expandonce\newterm@short s}%
      }%
      {%
        \DTLnewdbentry{\newterm@database}{ShortPlural}{\expandonce\newterm@shortplural}%
      }%
      \ifundef\newterm@longplural
      {%
        \DTLnewdbentry{\newterm@database}{LongPlural}{\expandonce\newterm@long s}%
      }%
      {%
        \DTLnewdbentry{\newterm@database}{LongPlural}{\expandonce\newterm@longplural}%
      }%
      \ifdefempty{\newterm@see}%
       {}%
       {\DTLnewdbentry{\newterm@database}{See}{\newterm@see}}%
      \ifdefempty{\newterm@seealso}%
       {}%
       {\DTLnewdbentry{\newterm@database}{SeeAlso}{\newterm@seealso}}%
```

Hook to make it easier to add extra fields.

```
      \newterm@extrafields
```

Add parent, if supplied.

```
      \ifdefempty{\newterm@parent}%
       {}%
       {%
         \iftermexists{\newterm@parent}%
         {%
           \edef\newterm@parentdatabase{\csuse{datagidxentry@\newterm@parent}}%
```

Parent entry must belong to same database as child entry.

```
           \ifthenelse{\equal{\newterm@parentdatabase}{\newterm@database}}
           {%
             \DTLnewdbentry{\newterm@database}{Parent}{\newterm@parent}%
             \datagidx@addchild{\newterm@database}{\newterm@parent}{\newterm@label}%
           }%
           {%
             \PackageError{datagidx}%
             {%
               Parent entry `\newterm@parent' must belong to the
               same database as child entry `\newterm@label'%
             }%
             {%
               Parent entry is in database
               `\newterm@parentdatabase' and child entry is in
               database `\newterm@database'%
             }%
           }%
         }%
         {%
           \PackageError{datagidx}%
           {%
              Can't assign parent to `\newterm@label':
              `\newterm@parent' doesn't exist%
           }%
           {}%
         }%
       }%
    }%
```

Provide user with a means to access the label of the latest defined term:

```
    \global\let\datagidxlastlabel\newterm@label
```

Allow user to hook in here

```
    \postnewtermhook
  }%
  %\end{macro}
  %
  %\begin{macro}{\postnewtermhook}
  %\changes{2.14}{2013-06-28}{new}
  %    \begin{macrocode}
  \newcommand*{\postnewtermhook}{}
```

\newtermfield Expandable access to field name. (No check for existence of the field. Uses etoolbox's \csuse, so expands to an empty string if the field is undefined.)

```
\newcommand*{\newtermfield}[1]{\csuse{newterm@#1}}
```

\ifnewtermfield If the named field (given in first argument) is empty or undefined do third argument, otherwise do second argument.

```
\newcommand{\ifnewtermfield}[3]{%
  \ifcsdef{newterm@#1}
  {%
    \ifcsempty{newterm@#1}{#3}{#2}%
  }%
  {%
    #3%
  }%
}
```

atagidx@newterm Normal behaviour for \newterm

```
\newcommand{\datagidx@newterm}[2][]{%
```

Assign values to all the fields.

```
  \datagidx@setfieldvalues{#1}{#2}%
```

Check if database exists.

```
  \DTLifdbexists{\newterm@database}%
  {%
```

Database exists. Check if term already exists.

```
    \iftermexists{\newterm@label}%
    {%
      \PackageError{datagidx}{Term `\newterm@label' already
        exists in database `\newterm@database'}{}%
    }%
    {%
```

Add this entry to the database.

```
      \datagidx@add@term{#2}%
    }%
  }%
  {%
```

Database doesn't exist.

```
    \PackageError{datagidx}%
    {Glossary/index data base `\newterm@database' doesn't exist}%
    {%
      You must define the glossary/index data base before you can
      add any terms to it.%
    }%
  }%
}
```

highopt@newterm Used when high optimized setting enabled. This setting must be switched off if the user wants to modify the database.

```
\newcommand{\datagidx@highopt@newterm}[2][]{%
```

Assign values to all the fields.

```
  \datagidx@setfieldvalues{#1}{#2}%
```

Check if database exists.

```
  \DTLifdbexists{\newterm@database}%
  {%
```

Database exists. If this is the first run, we need to add the term as usual, otherwise we just need to define \datagidxentry@⟨*label*⟩

```
    \edef\dtl@dogetrow{%
      \noexpand\dtlgetrowindex
      {\noexpand\dtl@rowidx}%
      {\newterm@database}%
      {%
        \dtlcolumnindex{\newterm@database}{Label}%
      }%
      {\newterm@label}}%
    \dtl@dogetrow
    \ifx\dtl@rowidx\dtlnovalue
```

Hasn't been defined so add.

```
       \datagidx@add@term{#2}%
```

Database will need to be sorted.

```
       \csdef{datagidx@do@highopt@sort@\newterm@database}{\datagidx@sort}%
    \else
```

Has been defined, so just define \datagidxentry@⟨*label*⟩ and \datagidxlastlabel

```
      \global\cslet{datagidxentry@\newterm@label}{\newterm@database}%
      \global\let\datagidxlastlabel\newterm@label
    \fi
  }%
  {%
```

Database doesn't exist.

```
    \PackageError{datagidx}%
    {Glossary/index data base `\newterm@database' doesn't exist}%
    {%
      You must define the glossary/index data base before you can
      add any terms to it.%
    }%
  }%
}
```

tagidx@addchild

```
\newcommand*{\datagidx@addchild}[3]{%
  \edef\dtl@dogetrow{%
```

```
      \noexpand\dtlgetrowforvalue
      {#1}%
      {%
        \dtlcolumnindex{\newterm@database}{Label}%
      }%
      {#2}}%
    \dtl@dogetrow
    \dtlgetentryfromcurrentrow
      {\datagidx@child}%
      {\dtlcolumnindex{#1}{Child}}%
    \ifx\datagidx@child\dtlnovalue
      \edef\datagidx@child{#3}%
    \else
      \edef\datagidx@child{\datagidx@child,#3}%
    \fi
    \edef\do@update{\noexpand\dtlupdateentryincurrentrow
      {Child}{\datagidx@child}}%
    \do@update
    \dtlrecombine
  }
```

5.5.3 Defining Acronyms

\newacro

\newacro[⟨*options*⟩]{⟨*short*⟩}{⟨*long*⟩}

Shortcut command for acronyms.

```
  \newcommand{\newacro}[3][]{%
    \newterm
      [%
        description={\capitalisewords{#3}},%
        short={\acronymfont{#2}},%
        long={#3},%
        text={\DTLgidxAcrStyle{#3}{\acronymfont{#2}}},%
        plural={\DTLgidxAcrStyle{#3s}{\acronymfont{#2s}}},%
        sort={#2},%
        #1%
      ]%
      {\MakeTextUppercase{#2}}%
  }
```

\acronymfont The font to use for the acronym.

```
  \newcommand*{\acronymfont}[1]{#1}
```

\DTLgidxAcrStyle

\DTLgidxAcrStyle{⟨*long*⟩}{⟨*short*⟩}

```
\newcommand*{\DTLgidxAcrStyle}[2]{#1 (#2)}
```

5.6 Conditionals

\iftermexists

```
\iftermexists{⟨label⟩}{⟨true part⟩}{⟨false part⟩}
```

Check if term with given label exists.

```
\newcommand{\iftermexists}[3]{%
  \ifcsdef{datagidxentry@#1}{#2}{#3}%
}
```

\datagidxdb Gets the label of database containing the given entry. No check is made for the existence of the entry. Expands to empty if label is undefined.

```
\newcommand*{\datagidxdb}[1]{%
  \csuse{datagidxentry@#1}%
}
```

\ifentryused

```
\ifentryused{⟨label⟩}{⟨true part⟩}{⟨false part⟩}
```

Check if entry with given label has been used.

```
\newcommand*{\ifentryused}[3]{%
  \letcs{\newterm@database}{datagidxentry@#1}%
```

\dtlgetrowforvalue doesn't expand the value when it checks for a match, so make sure label is fully expanded.

```
  \edef\dtl@dogetrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {%
      \dtlcolumnindex{\newterm@database}{Label}%
    }%
    {#1}}%
  \dtl@dogetrow
  \dtlgetentryfromcurrentrow
    {\datagidx@value}%
    {\dtlcolumnindex{\newterm@database}{Used}}%
  \ifnum\datagidx@value=1\relax
    #2%
  \else
    #3%
  \fi
}
```

5.7 Unsetting and Resetting

\glsreset \glsunset{⟨*label*⟩}

Mark as un-used.

```
\newcommand*{\glsreset}[1]{%
```

Fetch the name of the database with which this entry is associated.

```
  \letcs{\newterm@database}{datagidxentry@#1}%
```

Get the row associated with this label and make it the current row.

```
  \edef\do@getrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {\dtlcolumnindex{\newterm@database}{Label}}%
    {#1}%
  }%
  \do@getrow
```

Update the Used field.

```
  \dtlreplaceentryincurrentrow
    {0}{\dtlcolumnindex{\newterm@database}{Used}}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
  \dtlrecombine
}
```

\glsunset \glsunset{⟨*label*⟩}

Mark as used without affecting location.

```
\newcommand*{\glsunset}[1]{%
```

Fetch the name of the database with which this entry is associated.

```
  \letcs{\newterm@database}{datagidxentry@#1}%
```

Get the row associated with this label and make it the current row.

```
  \edef\do@getrow{%
    \noexpand\dtlgetrowforvalue
    {\newterm@database}%
    {\dtlcolumnindex{\newterm@database}{Label}}%
    {#1}%
  }%
  \do@getrow
```

Update the Used field.

```
  \dtlreplaceentryincurrentrow
    {1}{\dtlcolumnindex{\newterm@database}{Used}}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
    \dtlrecombine
  }
```

\glsresetall

```
\glsresetall{⟨db⟩}
```

Resets all entries in the given database.

```
  \newcommand*{\glsresetall}[1]{%
    \def\datagidx@list{}%
    \dtlforcolumn{\datagidx@label}{#1}{Label}%
    {%
      \ifdefempty\datagidx@list
      {%
        \let\datagidx@list\datagidx@label
      }%
      {%
        \eappto\datagidx@list{,\datagidx@label}%
      }%
    }%
    \@for\datagidx@thislabel:=\datagidx@list\do
    {%
      \glsreset{\datagidx@thislabel}%
    }%
  }
```

\glsunsetall

```
\glsunsetall{⟨db⟩}
```

Resets all entries in the given database.

```
  \newcommand*{\glsunsetall}[1]{%
    \def\datagidx@list{}%
    \dtlforcolumn{\datagidx@label}{#1}{Label}%
    {%
      \ifdefempty\datagidx@list
      {%
        \let\datagidx@list\datagidx@label
      }%
      {%
        \eappto\datagidx@list{,\datagidx@label}%
      }%
    }%
    \@for\datagidx@thislabel:=\datagidx@list\do
    {%
      \glsunset{\datagidx@thislabel}%
    }%
```

```
}
```

5.8 Accessing Entry Information

idx@anchorcount

Register to make unique anchors.

```
\newcount\datagidx@anchorcount
```

dx@formatanchor

Format number using six digits.

```
\newcommand*{\datagidx@formatanchor}[1]{%
  \ifnum#1<10000
   0%
    \ifnum#1<1000
     0%
      \ifnum#1<100
       0%
        \ifnum#1<10
         0%
        \fi
      \fi
    \fi
  \fi
  \number#1%
}
```

datagidx@escloc

```
\newcommand*{\@datagidx@escloc}[2]{%
  \expandafter\string\csname#1\endcsname{\noexpand\number#2}%
}
```

@escapelocation

```
\newcommand*{\datagidx@escapelocation}{%
  \def\@arabic{\@datagidx@escloc{@arabic}}%
  \def\@roman{\@datagidx@escloc{@roman}}%
  \def\@Roman{\@datagidx@escloc{@Roman}}%
  \def\@alph{\@datagidx@escloc{@alph}}%
  \def\@Alph{\@datagidx@escloc{@Alph}}%
}
```

elocationformat

```
\newcommand*{\datagidx@escapelocationformat}{%
  \def\@arabic##1{arabic}%
  \def\@roman##1{roman}%
  \def\@Roman##1{Roman}%
  \def\@alph##1{alph}%
  \def\@Alph##1{Alph}%
}
```

rlocationformat

```
\newcommand*{\datagidx@clearlocationformat}{%
  \let\@arabic\@firstofone
  \let\@roman\@firstofone
  \let\@Roman\@firstofone
  \let\@alph\@firstofone
  \let\@Alph\@firstofone
}
```

AddLocationType

Allow user to add their own location type. Argument must be control sequence name without initial backslash.

```
\newcommand*{\DTLgidxAddLocationType}[1]{%
  \gappto\datagidx@escapelocation{%
    \expandafter\def\csname#1\endcsname{\@datagidx@escloc{#1}}%
  }%
  \gappto\datagidx@escapelocationformat{%
    \expandafter\def\csname#1\endcsname##1{#1}%
  }%
  \gappto\datagidx@clearlocationformat{%
    \expandafter\let\csname#1\endcsname\@firstofone
  }%
}
```

May only be used in the preamble. (Needs to be set before the aux file is read.)

```
\@onlypreamble\DTLgidxAddLocationType
```

\datagidx@target

\datagidx@target{⟨*label*⟩}{⟨*format*⟩}{⟨*location*⟩}{⟨*text*⟩}

Make a target if \hypertarget has been defined.

```
\newcommand*{\datagidx@target}[4]{%
  \global\advance\datagidx@anchorcount by 1\relax
  \edef\@datagidx@target{datagidx.\datagidx@formatanchor\datagidx@anchorcount}%
  \ifstrempty{#3}
  {%
    \datagidx@write@usedentry{#1}{}%
  }%
  {%
    \bgroup
      \datagidx@escapelocation
```

Need to prevent \@arabic etc from being expanded just yet (or it will throw the page numbering out of sync for entries that occur by a page break).

```
      \def\@arabic{\noexpand\@arabic}%
      \def\@roman{\noexpand\@roman}%
      \def\@Roman{\noexpand\@Roman}%
      \def\@alph{\noexpand\@alph}%
      \def\@Alph{\noexpand\@Alph}%
```

```
        \protected@edef\@datagidx@dowriteaux{%
          \noexpand\datagidx@write@usedentry{#1}%
               {[#2]{#3}{\@datagidx@target}}%
        }%
        \@datagidx@dowriteaux
      \egroup
    }%
    \ifdef\hypertarget
    {%
```

Make sure the current line doesn't scroll off the top of the screen.

```
      \datagidxshowifdraft
      {%
        [\@datagidx@target]%
        \discretionary{}{}{}%
      }%
      \bgroup
        \let\glsadd\@gobble
        \settoheight\dimen@{#4}%
        \raisebox{\dimen@}%
        {%
          \datagidxtarget{\@datagidx@target}{}%
        }%
      \egroup
    }%
    {%
    }%
    \datagidxshowifdraft{[#1]\discretionary{}{}{}}%
    #4%
  }
```

\glsdispentry

```
\glsdispentry{⟨label⟩}{⟨field⟩}
```

Short cut that fetches and displays a value.

```
  \DeclareRobustCommand*{\glsdispentry}[2]{%
    \DTLgidxFetchEntry{\datagidx@dispentryval}{#1}{#2}%
    \datagidx@dispentryval
  }
```

\Glsdispentry

```
\Glsdispentry{⟨label⟩}{⟨field⟩}
```

As previous but makes the first letter upper case.

```
  \DeclareRobustCommand*{\Glsdispentry}[2]{%
    \DTLgidxFetchEntry{\datagidx@dispentryval}{#1}{#2}%
    \xmakefirstuc\datagidx@dispentryval
  }
```

LgidxFetchEntry Fetch value for the given field for the term identified by ⟨*label*⟩ and store the value in ⟨*cs*⟩ (a control sequence).

```
  \newcommand*{\DTLgidxFetchEntry}[3]{%
```

Does this entry exist?

```
    \ifcsdef{datagidxentry@#2}%
    {%
```

Fetch the name of the database with which this entry is associated.

```
    \letcs{\newterm@database}{datagidxentry@#2}%
```

Get the row associated with this label and make it the current row.

```
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\newterm@database}%
      {\dtlcolumnindex{\newterm@database}{Label}}%
      {#2}%
    }%
    \do@getrow
```

Get the entry for the given field in the current row and store in ⟨*cs*⟩.

```
    \dtlgetentryfromcurrentrow
      {#1}%
      {\dtlcolumnindex{\newterm@database}{#3}}%
    }%
    {%
```

Entry hasn't been defined.

```
      \PackageError{datagidx}{No term ‘#2’ defined}{}%
    }%
  }
```

arse@formatlabel `\parse@formatlabel{⟨[⟨format⟩]label⟩}`

Separate format and label from argument.

```
  \newcommand*{\datagidx@parse@formatlabel}[1]{%
    \datagidx@parse@format@label@#1\@endparse@formatlabel@
  }
  \newcommand*\datagidx@parse@format@label@{%
    \@ifnextchar[{\datagidx@parse@formatlabel@}{\datagidx@parse@formatlabel@[]}%
  }
  \def\datagidx@parse@formatlabel@[#1]#2\@endparse@formatlabel@{%
    \def\datagidx@format{#1}%
    \def\datagidx@label{#2}%
  }
```

tagidx@use@entry

\@datagidx@use@entry{⟨*link text*⟩}

The label and format should have been stored in `\datagidx@label` and `\datagidx@format` before calling this macro.

```
  \newcommand*{\@datagidx@use@entry}[1]{%
```

Does this term exist?

```
    \ifcsundef{datagidxentry@\datagidx@label}
    {%
      \PackageError{datagidx}{Entry `\datagidx@label' doesn't exist}{}%
    }%
    {%
```

Fetch the name of the database with which this entry is associated.

```
      \letcs{\newterm@database}{datagidxentry@\datagidx@label}%
```

Get the row associated with this label and make it the current row.

```
      \edef\do@getrow{%
        \noexpand\dtlgetrowforvalue
        {\newterm@database}%
        {\dtlcolumnindex{\newterm@database}{Label}}%
        {\datagidx@label}%
      }%
      \do@getrow
```

Get the entry for the `FirstId` field and store in `\datagidx@id`

```
      \dtlgetentryfromcurrentrow
        {\datagidx@id}%
        {\dtlcolumnindex{\newterm@database}{FirstId}}%
```

If it hasn't been defined set it.

```
      \DTLifnull\datagidx@id
      {%
```

Count register hasn't been updated yet.

```
        \count@=\datagidx@anchorcount\relax
        \advance\count@ by 1\relax
        \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\count@}%
      }%
      {}%
```

Update the `Used` field.

```
      \dtlreplaceentryincurrentrow
        {1}{\dtlcolumnindex{\newterm@database}{Used}}%
```

Get the parent entry label (if one exists).

```
      \dtlgetentryfromcurrentrow
        {\datagidx@parent}%
        {\dtlcolumnindex{\newterm@database}{Parent}}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
      \dtlrecombine
```

If parent hasn't be used, give it an empty location.

```
      \datagidx@markparent{\newterm@database}{\datagidx@parent}%
```

Write the location to the auxiliary file and display value of field.

```
      \datagidx@target{\datagidx@label}{\datagidx@format}%
        {\csuse{the\DTLgidxCounter}}{#1}%
   }%
  }
```

\DTLgidxCounter The counter used for the location lists.

```
  \newcommand*{\DTLgidxCounter}{page}
```

gidx@markparent Assign empty location to parent, if location of that parent is null. (Recursive).

```
  \newcommand*{\datagidx@markparent}[2]{%
    \ifx#2\dtlnovalue
```

Null parent, so break out of recursion.

```
    \else
```

Write empty location to the auxiliary file.

```
      \datagidx@target{#2}{}{}{}%
```

Fetch this parent's parent entry. Get the row associated with this and make it the current row.

```
      \edef\do@getrow{%
        \noexpand\dtlgetrowforvalue
        {#1}%
        {\dtlcolumnindex{#1}{Label}}%
        {#2}}%
      \do@getrow
```

Get the entry for the FirstId field and store in \datagidx@id

```
    \dtlgetentryfromcurrentrow
      {\datagidx@id}%
      {\dtlcolumnindex{\newterm@database}{FirstId}}%
```

If it hasn't been defined set it.

```
    \DTLifnull\datagidx@id
    {%
      \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount}%
    }%
    {}%
```

Get the parent

```
      \dtlgetentryfromcurrentrow
        {\datagidx@parent}%
        {\dtlcolumnindex{#1}{Parent}}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
    \dtlrecombine
```

Recurse

```
      \datagidx@markparent{#1}{\datagidx@parent}%
    \fi
  }
```

write@usedentry Write out location to aux file and add location to the location list for the current run.

```
  \newcommand*{\datagidx@write@usedentry}[2]{%
```

Do update if 'highopt optimize' setting is on.

```
    \datagidx@do@highopt@update{#1}%
```

Write out location to aux file.

```
    \protected@write{\@auxout}{}%
      {%
        \string\datagidx@usedentry{#1}{#2}%
      }%
```

Add to this run's location field.

```
    \protected@edef\datagidx@do@usedentry{%
      \noexpand\datagidx@xusedentry{CurrentLocation}{#1}{#2}%
    }%
```

If the location counter is the page counter, defer until after the page break.

```
    \expandafter\ifstrequal\expandafter{\DTLgidxCounter}{page}%
    {%
       \expandafter\afterpage\expandafter{\datagidx@do@usedentry}%
    }%
    {
      \datagidx@do@usedentry
    }%
  }
```

agidx@xusedentry

```
\datagidx@usedentry{⟨location tag⟩}{⟨label⟩}{⟨location⟩}
```

Like \datagidx@usedentry but expands the location. Unlike \datagidx@usedentry the first argument isn't optional.

```
  \newcommand*{\datagidx@xusedentry}[3]{%
    \protected@edef\@datagidx@do@xusedentry{%
       \noexpand\datagidx@usedentry[#1]{#2}{#3}%
    }%
    \@datagidx@do@xusedentry
  }
```

tagidx@usedentry

```
\datagidx@usedentry[⟨location tag⟩]{⟨label⟩}{⟨location⟩}
```

Add to the location list for the given entry.

```
\newcommand*{\datagidx@usedentry}[3][Location]{%
```

Check if label exists. (It may have been deleted or had a label change.)

```
  \ifcsundef{datagidxentry@#2}%
  {%
    \PackageWarning{datagidx}{No term '#2' defined. Ignoring}%
  }%
  {%
% Fetch the name of the database with which this entry is
% associated.
%    \begin{macrocode}
    \letcs{\newterm@database}{datagidxentry@#2}%
```

Get the row associated with this label and make it the current row.

```
    \edef\do@getrow{%
      \noexpand\dtlgetrowforvalue
      {\newterm@database}%
      {\dtlcolumnindex{\newterm@database}{Label}}%
      {#2}%
    }%
    \do@getrow
```

```
% Get the entry for the \meta{location tag} field in the current row and store in
% \cs{datagidx@loc}.
%    \begin{macrocode}
    \dtlgetentryfromcurrentrow
      {\datagidx@loc}%
      {\dtlcolumnindex{\newterm@database}{#1}}%
```

Check the success of the previous command.

```
    \ifx\datagidx@loc\dtlnovalue
```

There's no ⟨*location tag*⟩ field in the current row, so add one with the given location.

```
      \def\datagidx@loc{#3}%
      \dtlappendentrytocurrentrow{#1}{\expandonce\datagidx@loc}%
    \else
```

There is a ⟨*location tag*⟩ field in the current row, so append the given location to the list, unless one or the other is empty.

```
      \ifdefempty{\datagidx@loc}%
      {%
        \def\datagidx@loc{#3}%
      }%
      {%
        \ifstrempty{#3}%
        {}%
        {%
          \appto\datagidx@loc{,#3}%
        }%
      }%
```

and update the entry in the current row.

```
        \expandafter\dtlreplaceentryincurrentrow\expandafter
          {\datagidx@loc}%
          {\dtlcolumnindex{\newterm@database}{#1}}%
      \fi
```

Current row has been edited, so we need to merge the current row back into the database.

```
      \dtlrecombine
    }%
  }
```

tagidx@save@loc Store the current location from the previous run.

```
  \newcommand*{\datagidx@save@loc}[2]{%
   \bgroup
    \datagidx@escapelocation
    \xdef\datagidx@tmp{#2}%
   \egroup
   \expandafter\xdef\csname datagidx@prev@loc@#1\endcsname{\datagidx@tmp}%
  }
```

\glsadd

\glsadd{⟨*[⟨format⟩]label*⟩}

```
  \newcommand*{\glsadd}[1]{%
    \NoCaseChange{\@glsadd{#1}}%
  }
  \DeclareRobustCommand*{\@glsadd}[1]{%
```

Check term has been defined.

```
    \ifcsundef{datagidxentry@\datagidx@label}%
    {%
      \PackageError{datagidx}{Term ‘\datagidx@label’ doesn’t exist}{}%
    }%
    {%

      \datagidx@parse@formatlabel{#1}%
```

Write the location to the auxiliary file.

```
      \datagidx@target{\datagidx@label}{\datagidx@format}%
        {\csuse{the\DTLgidxCounter}}{}%
```

Fetch the name of the database with which this entry is associated.

```
      \letcs{\newterm@database}{datagidxentry@\datagidx@label}%
```

Get the row associated with this label and make it the current row.

```
      \edef\do@getrow{%
        \noexpand\dtlgetrowforvalue
        {\newterm@database}%
        {\dtlcolumnindex{\newterm@database}{Label}}%
        {\datagidx@label}%
      }%
      \do@getrow
```

Update the Used field.

```
    \dtlreplaceentryincurrentrow
      {1}{\dtlcolumnindex{\newterm@database}{Used}}%
```

Get the entry for the FirstId field and store in \datagidx@id

```
    \dtlgetentryfromcurrentrow
      {\datagidx@id}%
      {\dtlcolumnindex{\newterm@database}{FirstId}}%
```

If it hasn't been defined set it.

```
    \DTLifnull\datagidx@id
    {%
      \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount}%
    }%
    {}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
    \dtlrecombine
  }%
}
```

\datagidx@count Loop counter used by \glsaddall

```
\newcount\datagidx@count
```

\glsaddall \glsaddall{⟨*db*⟩}

Adds all entries in the given database.

```
\newcommand*{\glsaddall}[1]{%
  \DTLifdbexists{#1}%
  {%
     \edef\datagidx@rowcount{\number\DTLrowcount{#1}}%
     \datagidx@count=0\relax
     \loop
       \advance\datagidx@count by 1\relax
       \dtlgetrow{#1}{\datagidx@count}%
```

Get the label for this row.

```
       \dtlgetentryfromcurrentrow
         {\datagidx@label}%
         {\dtlcolumnindex{#1}{Label}}%
```

Write blank location to the auxiliary file but temporarily undefine \hypertarget as it doesn't make sense to have a target here.

```
    \bgroup
      \undef\hypertarget
      \datagidx@target{\datagidx@label}{}{}{}%
    \egroup
```

Update the Used field.

```
       \dtlreplaceentryincurrentrow
        {1}{\dtlcolumnindex{#1}{Used}}%
```

Get the entry for the FirstId field and store in \datagidx@id

```
      \dtlgetentryfromcurrentrow
        {\datagidx@id}%
        {\dtlcolumnindex{#1}{FirstId}}%
```

If it hasn't been defined set it.

```
      \DTLifnull\datagidx@id
      {%
        \dtlappendentrytocurrentrow{FirstId}{\datagidx@formatanchor\datagidx@anchorcount}%
      }%
      {}%
```

Current row has been edited, so we need to merge the current row back into the database.

```
      \dtlrecombine
```

Repeat loop if not finished.

```
     \ifnum\datagidx@count<\datagidx@rowcount
     \repeat
  }%
  {%
    \PackageError{datagidx}{Database ‘#1’ doesn’t exist}{}%
  }%
}
```

\glslink

```
\glslink{⟨label⟩}{⟨text⟩}
```

Use given entry but user supplies text.

```
\DeclareRobustCommand*{\glslink}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \datagidxlink{\datagidx@label}%
  {%
    \@datagidx@use@entry{#2}%
  }%
}
```

\useentry

```
\useentry{⟨label⟩}{⟨field⟩}
```

Fetch and use the given field for the given entry.

```
\DeclareRobustCommand*{\useentry}[2]{%
```

```
    \datagidx@parse@formatlabel{#1}%
    \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
    \datagidxlink{\datagidx@label}%
    {%
      \@datagidx@use@entry{\datagidx@value}%
    }%
  }
```

\Useentry

```
\Useentry{⟨label⟩}{⟨field⟩}
```

As \useentry, but capitalise the first word.

```
  \DeclareRobustCommand*{\Useentry}[2]{%
    \datagidx@parse@formatlabel{#1}%
    \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
    \datagidxlink{\datagidx@label}%
    {%
      \@datagidx@use@entry{\xmakefirstuc{\datagidx@value}}%
    }%
  }
```

\USEentry

```
\USEentry{⟨label⟩}{⟨field⟩}
```

As \useentry, but make the whole term upper case.

```
  \DeclareRobustCommand*{\USEentry}[2]{%
    \datagidx@parse@formatlabel{#1}%
    \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
    \datagidxlink{\datagidx@label}%
    {%
      \@datagidx@use@entry{\MakeTextUppercase{\datagidx@value}}%
    }%
  }
```

\useentrynl

```
\useentrynl{⟨label⟩}{⟨field⟩}
```

Fetch and use the given field for the given entry without creating a hyperlink.

```
  \DeclareRobustCommand*{\useentrynl}[2]{%
    \datagidx@parse@formatlabel{#1}%
    \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
    \@datagidx@use@entry{\datagidx@value}%
  }
```

\Useentrynl

```
\Useentrynl{⟨label⟩}{⟨field⟩}
```

As \useentry, but capitalise the first word.

```
\DeclareRobustCommand*{\Useentrynl}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \@datagidx@use@entry{\xmakefirstuc{\datagidx@value}}%
}
```

\USEentrynl

```
\USEentrynl{⟨label⟩}{⟨field⟩}
```

As \useentry, but make the whole term upper case.

```
\DeclareRobustCommand*{\USEentrynl}[2]{%
  \datagidx@parse@formatlabel{#1}%
  \DTLgidxFetchEntry{\datagidx@value}{\datagidx@label}{#2}%
  \@datagidx@use@entry{\MakeTextUppercase{\datagidx@value}}%
}
```

Short cuts to common fields.

\gls

```
\DeclareRobustCommand*{\gls}[1]{\useentry{#1}{Text}}
```

\glspl

```
\DeclareRobustCommand*{\glspl}[1]{\useentry{#1}{Plural}}
```

\Gls

```
\DeclareRobustCommand*{\Gls}[1]{\Useentry{#1}{Text}}
```

\Glspl

```
\DeclareRobustCommand*{\Glspl}[1]{\Useentry{#1}{Plural}}
```

\glsnl

```
\DeclareRobustCommand*{\glsnl}[1]{\useentrynl{#1}{Text}}
```

\glsplnl

```
\DeclareRobustCommand*{\glsplnl}[1]{\useentrynl{#1}{Plural}}
```

\Glsnl

```
\DeclareRobustCommand*{\Glsnl}[1]{\Useentrynl{#1}{Text}}
```

\Glsplnl

```
\DeclareRobustCommand*{\Glsplnl}[1]{\Useentrynl{#1}{Plural}}
```

\glssym

```
\DeclareRobustCommand*{\glssym}[1]{\useentry{#1}{Symbol}}
```

\Glssym

```
\DeclareRobustCommand*{\Glssym}[1]{\Useentry{#1}{Symbol}}
```

5.8.1 Using Acronyms

DTLgidxFormatAcr

```
\DTLgidxFormatAcr{⟨label⟩}{⟨long field⟩}{⟨short field⟩}
```

```
\newcommand*{\DTLgidxFormatAcr}[3]{%
  \DTLgidxAcrStyle{\glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
```

LgidxFormatAcrUC

```
\DTLgidxFormatAcr{⟨label⟩}{⟨long field⟩}{⟨short field⟩}
```

As previous but capitalise first word.

```
\newcommand*{\DTLgidxFormatAcrUC}[3]{%
  \DTLgidxAcrStyle{\Glsdispentry{#1}{#2}}{\useentry{#1}{#3}}%
}
```

\acr

```
\DeclareRobustCommand*{\acr}[1]{%
  \ifentryused{#1}%
  {\useentry{#1}{Short}}%
  {\DTLgidxFormatAcr{#1}{Long}{Short}}%
}
```

\acrpl

```
\DeclareRobustCommand*{\acrpl}[1]{%
  \ifentryused{#1}%
  {\useentry{#1}{ShortPlural}}%
  {\DTLgidxFormatAcr{#1}{LongPlural}{ShortPlural}}%
}
```

\Acr

```
\DeclareRobustCommand*{\Acr}[1]{%
  \ifentryused{#1}%
  {\Useentry{#1}{Short}}%
  {\DTLgidxFormatAcrUC{#1}{Long}{Short}}%
}
```

\Acrpl

```
\DeclareRobustCommand*{\Acrpl}[1]{%
  \ifentryused{#1}%
  {\Useentry{#1}{ShortPlural}}%
  {\DTLgidxFormatAcrUC{#1}{LongPlural}{ShortPlural}}%
}
```

5.9 Displaying Glossaries, Lists of Acronyms, Indices

Define keys for \printterms:

```
\define@key{printterms}{database}{\renewcommand*{\newterm@database}{#1}}
```

Options for post description.

```
\define@choicekey{printterms}{postdesc}[\val\nr]%
 {none,dot}%
 {%
   \datagidx@setpostdesc\nr
 }
```

Options for pre-location.

```
\define@choicekey{printterms}{prelocation}[\val\nr]%
 {none,enspace,space,dotfill,hfill}%
 {%
   \datagidx@setprelocation\nr
 }
```

How to display the location list.

```
\define@choicekey{printterms}{location}[\val\nr]%
 {hide,list,first}%
 {\datagidx@setlocation\nr}
```

How to format the symbol in relation to the description.

```
\define@choicekey{printterms}{symboldesc}[\val\nr]%
 {symbol,desc,(symbol) desc,desc (symbol),symbol desc,desc symbol}%
 {\datagidx@formatsymdesc\nr}
```

How many columns to have.

```
\define@key{printterms}{columns}%
{%
  \DTLgidxSetColumns{#1}%
}
```

How to format the name.

```
\define@choicekey{printterms}{namecase}[\val\nr]%
 {nochange,uc,lc,firstuc,capitalise}%
 {%
   \datagidx@setnamecase\nr
 }
```

```
\define@key{printterms}{namefont}%
{%
  \renewcommand*{\DTLgidxNameFont}[1]{{#1{##1}}}%
}
```

```
\define@key{printterms}{postname}
{%
  \renewcommand*{\DTLgidxPostName}{#1}%
}
```

```
\define@choicekey{printterms}{see}[\val\nr]%
  {comma,brackets,dot,space,nosep,semicolon,location}%
  {\datagidx@setsee\nr}
```

```
\define@choicekey{printterms}{child}[\val\nr]%
 {named,noname}%
 {%
   \datagidx@setchildstyle\nr
 }
```

Symbol width

```
\define@key{printterms}{symbolwidth}%
{%
  \setlength{\datagidxsymbolwidth}{#1}%
}
```

Location width

```
\define@key{printterms}{locationwidth}%
{%
  \setlength{\datagidxlocationwidth}{#1}%
}
```

Child sort:

```
\define@choicekey{printterms}{childsort}[\val\nr]%
 {true,false}[true]%
 {%
   \datagidx@setchildsort\nr
 }
```

Change style:

```
\define@choicekey{printterms}{showgroups}{true,false}[true]{%
 \appto\newterm@styles{showgroups={#1},}%
}
```

```
\define@key{printterms}{style}{\appto\newterm@styles{style={#1},}}
```

```
\define@key{printterms}{heading}{\appto\newterm@styles{heading={#1},}}
```

```
\define@key{printterms}{postheading}{%
  \appto\newterm@styles{postheading={#1},}%
}
```

```
\define@key{printterms}{sort}{\appto\newterm@styles{sort={#1},}}
```

```
\define@choicekey{printterms}{balance}[\val\nr]{true,false}[true]{%
  \ifcase\nr\relax
    \appto\newterm@styles{balance=true,}%
  \or
    \appto\newterm@styles{balance=false,}%
  \fi
}
```

terms@condition

```
\newcommand*{\printterms@condition}{\boolean{true}}
\define@key{printterms}{condition}{\renewcommand*{\printterms@condition}{#1}}
```

nttermsstartpar

```
\newcommand{\printtermsstartpar}{\par}
```

s@setupmulticol

```
\newcommand*{\printterms@setupmulticol}{%
 \ifdefempty\datagidx@postheading
 {%
   \edef\datagidx@prestart{%
    \noexpand\datagidx@heading{\noexpand\datagidx@title}%
    \noexpand\begin{\datagidx@multicols}{\datagidx@columns}%
   }%
 }%
 {%
   \edef\datagidx@prestart{%
    \noexpand\datagidx@heading{\noexpand\datagidx@title}%
    \noexpand\begin{\datagidx@multicols}{\datagidx@columns}%
    [\noexpand\datagidx@postheading]%
   }%
 }%
 \edef\datagidx@postend{%
   \noexpand\end{\datagidx@multicols}%
 }%
}
```

rms@setuptwocol

```
\newcommand*{\printterms@setuptwocol}{%
 \def\datagidx@prestart{%
   \twocolumn[\datagidx@heading{\datagidx@title}%
   \datagidx@postheading]}%
 \if@twocolumn
   \def\datagidx@postend{}%
 \else
   \def\datagidx@postend{\printtermsrestoreonecolumn}%
 \fi
}
```

estoreonecolumn

```
\newcommand{\printtermsrestoreonecolumn}{\onecolumn}
```

\printterms

```
\printterms[options]
```

Print the list of terms

```
\newcommand{\printterms}[1][]{%
\bgroup
```

Set default database.

```
  \let\newterm@database\datagidx@defaultdatabase
```

Initialise key list for style:

```
  \let\newterm@styles\@empty
```

Set options:

```
  \setkeys{printterms}{#1}%
```

Check if database exists.

```
  \DTLifdbexists{\newterm@database}%
  {%
```

Provide user the means to access the current database name.

```
    \edef\DTLgidxCurrentdb{\newterm@database}%
```

Get the fields from datagidx:

```
    \edef\do@getrow{\noexpand\dtlgetrowforvalue
      {datagidx}%
      {\dtlcolumnindex{datagidx}{Glossary}}%
      {\newterm@database}%
    }%
    \do@getrow
    \dtlgetentryfromcurrentrow
      {\datagidx@title}%
      {\dtlcolumnindex{datagidx}{Title}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@heading}%
      {\dtlcolumnindex{datagidx}{Heading}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@postheading}%
      {\dtlcolumnindex{datagidx}{PostHeading}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@multicols}%
      {\dtlcolumnindex{datagidx}{MultiCols}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@sort}%
      {\dtlcolumnindex{datagidx}{Sort}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@style}%
      {\dtlcolumnindex{datagidx}{Style}}%
    \dtlgetentryfromcurrentrow
      {\datagidx@showgroups}%
      {\dtlcolumnindex{datagidx}{ShowGroups}}%
```

Allow user to override style here.

```
\edef\dtl@do@setkeys{\noexpand\setkeys{newgloss}{\expandonce\newterm@styles}}%
\dtl@do@setkeys
```

Do we need to use multicols?

```
\ifnum\datagidx@columns>1\relax
   \ifnum\datagidx@columns=2\relax
    \ifdatagidxbalance
      \printterms@setupmulticol
    \else
      \printterms@setuptwocol
    \fi
   \else
      \printterms@setupmulticol
   \fi
\else
   \def\datagidx@prestart{}%
   \def\datagidx@postend{}%
\fi
\let\@dtl@dbname\DTLgidxCurrentdb
```

Set the style

```
\csuse{datagidxshowgroups\datagidx@showgroups}%
\datagidxsetstyle{\datagidx@style}%
```

Now display the glossary/index:

```
\def\datagidx@labellist{}%
\ifnum\datagidx@columns=1\relax
  \datagidx@heading{\datagidx@title}%
  \datagidx@postheading
\fi
\datagidx@do@sort
\datagidx@prestart
```

:

```
    \printtermsstartpar
    \datagidxstart
    \let\DTLgidxName\datagidx@invert
    \let\DTLgidxPlace\datagidx@invert
    \let\DTLgidxSubject\datagidx@invert
    \let\DTLgidxOffice\datagidx@invert
      \DTLgidxForeachEntry
      {%
         \datagidxitem
      }%
    \datagidxend
    \datagidx@postend
  }%
  {%
```

Database doesn’t exist.

```
      \PackageError{datagidx}%
      {Glossary/index data base `\newterm@database' doesn't exist}%
      {%
        You must define the glossary/index data base before you can
        use it.%
      }%
    }%
  \egroup
  }
```

tagidx@getgroup Get the current group.

```
  \def\datagidx@getgroup#1#2\datagidx@endgetgroup{%
    \dtl@setcharcode{#1}{\count@}%
    \dtlifintclosedbetween{\count@}{48}{57}%
    {%
      \gdef\datagidxcurrentgroup{Numbers}%
    }%
    {%
      \dtlifintclosedbetween{\count@}{97}{122}%
      {%
        \advance\count@ by -96\relax
        \xdef\datagidxcurrentgroup{\@Alph\count@}%
      }%
      {%
        \dtlifintclosedbetween{\count@}{65}{90}%
        {%
          \gdef\datagidxcurrentgroup{#1}%
        }%
        {%
          \gdef\datagidxcurrentgroup{Symbols}%
        }%
      }%
    }%
  }
```

roupHeaderTitle Produce the group title from the group label.

```
  \newcommand*{\DTLgidxGroupHeaderTitle}[1]{%
    \ifcsdef{datagidx#1name}
    {%
      \csuse{datagidx#1name}%
    }%
    {%
      #1%
    }%
  }
```

gidxForeachEntry

```
\DTLgidxForeachEntry{⟨body⟩}
```

Iterate through the current database, but only do ⟨*body*⟩ if there is a location or cross-reference.

```
\newcommand{\DTLgidxForeachEntry}[1]{%
  \def\datagidxprevgroup{}%
  \edef\datagidx@doforeachentry{%
    \noexpand\DTLforeach*[\expandonce\printterms@condition]{\DTLgidxCurrentdb}%
     {\expandonce\DTLgidxAssignList}
  }%
  \datagidx@doforeachentry
  {%
```

Iterate through top-level entries.

```
    \DTLifnull{\Parent}%
    {%
```

If there's no location, but there is a current location, then document needs updating.

```
      \DTLifnull\Location
      {%
         \DTLifnull\CurrentLocation
         {%
         }%
         {%
```

We have a current location but not a location.

```
           \global\let\@datagidx@dorerun@warn\@data@rerun@warn
         }%
      }%
      {%
```

We have a location. Is it up-to-date?

```
        \ifcsdef{datagidx@prev@loc@\Label}%
        {%
```

Current location was saved in the previous run. Has it changed? This may cause a problem on page boundaries so this code block may be removed in future versions.

```
          \dtlgidx@checklocationchange
        }%
        {%
```

Current location wasn't saved last run, so rerun required.

```
           \global\let\@datagidx@dorerun@warn\@data@rerun@warn
        }%
      }%
      \datagidx@doifdisplayed
      {%
```

Write current location to file to compare current and previous lists. (Can't compare `\Location` with `\CurrentLocation` as there may be locations occurring across a page boundary.)

```
        \edef\datagidx@dowrite{%
          \noexpand\protected@write\noexpand\@auxout{}%
          {%
```

```
            \string\datagidx@save@loc{\Label}{\CurrentLocation}%
          }%
        }%
        \datagidx@dowrite
```

Initialise level.

```
        \datagidx@level=1\relax
        \expandafter\datagidx@getgroup\Sort{}\datagidx@endgetgroup
        #1%
        \global\let\datagidxprevgroup\datagidxcurrentgroup
      }%
    }%
    {}%
  }%
}
```

klocationchange

```
\newcommand*{\dtlgidx@checklocationchange}{%
  \protected@edef\@prev@location{%
     \csname datagidx@prev@loc@\Label\endcsname}%
  \@onelevel@sanitize\@prev@location
  \protected@edef\@cur@location{\CurrentLocation}%
  \@onelevel@sanitize\@cur@location
  \ifdefequal{\@prev@location}{\@cur@location}%
  {}%
  {%
    \global\let\@datagidx@dorerun@warn\@data@rerun@warn
  }%
}
```

dx@doifdisplayed

\datagidx@doifdisplayed{⟨*body*⟩}

Do ⟨*body*⟩ if entry should appear in the glossary/index. \Location, \See and \SeeAlso must be set before use.

```
\newcommand{\datagidx@doifdisplayed}[1]{%
  \DTLifnull{\Location}%
  {%
     \DTLifnull{\See}
     {%
       \DTLifnull{\SeeAlso}{}%
       {%
         #1%
       }%
     }%
     {%
```

See is not null, but have any of the cross-referenced items been used?

```
       \@for\dtl@thislabel:=\See\do
       {%
```

Does the cross-referenced term exist?

```
           \iftermexists{\dtl@thislabel}%
           {%
```

Has it been used?

```
             \ifentryused{\dtl@thislabel}%
             {%
               #1%
```

Break out of loop.

```
               \@endfortrue
             }%
             {}%
           }%
           {%
           }%
       }%
     }%
  }%
  {%
    #1%
  }%
}%
```

\datagidx@level Keep track of current level

```
\newcount\datagidx@level
```

6 databib.sty

6.1 Package Declaration

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{databib}[2019/09/27 v2.32 (NLCT)]
```

Load required packages:

```
\RequirePackage{datatool}
```

6.2 Package Options

\dtlbib@style The default bib style is stored in \dtlbib@style.

```
\newcommand*{\dtlbib@style}{plain}
```

The style=databib package option sets \dtlbib@style.

```
\define@choicekey{databib.sty}{style}{plain,abbrv,alpha}{%
\def\dtlbib@style{#1}}
```

Process package options:

```
\ProcessOptionsX
```

6.3 Loading BBL file

\DTLloadbbl

```
\DTLloadbib[⟨bbl file⟩]{⟨db name⟩}{⟨bib list⟩}
```

```
\newcommand*{\DTLloadbbl}[3][\jobname.bbl]{%
\bibliographystyle{databib}%
\if@filesw
 \immediate\write\@auxout{\string\bibdata{#3}}%
\fi
\DTLnewdb{#2}%
\edef\DTLBIBdbname{#2}%
\@input@{#1}}
```

\DTLnewbibrow \DTLnewbibrow adds a new row to the bibliography database. (\DTLBIBdbname must be set prior to use to the name of the datatool database which must exist. Any check to determine its existence should be performed when \DTLBIBdbname is set.)

```
\newcommand*{\DTLnewbibrow}{\@DTLnewrow{\DTLBIBdbname}}
```

\DTLnewbibitem

```
\DTLnewbibitem{⟨key⟩}{⟨value⟩}
```

Adds a new database entry with the given key and value.

```
\newcommand*{\DTLnewbibitem}[2]{%
  \@DTLnewdbentry{\DTLBIBdbname}{#1}{#2}}
```

6.4 Predefined text

\andname

```
\providecommand*{\andname}{and}
```

\ofname

```
\providecommand*{\ofname}{of}
```

\inname

```
\providecommand*{\inname}{in}
```

\etalname

```
\providecommand*{\etalname}{et al.}
```

\editorname

```
\providecommand*{\editorname}{editor}
```

\editorsname

```
\providecommand*{\editorsname}{editors}
```

\volumename

```
\providecommand*{\volumename}{volume}
```

\numbername

```
\providecommand*{\numbername}{number}
```

\pagesname

```
\providecommand*{\pagesname}{pages}
```

\pagename

```
\providecommand*{\pagename}{page}
```

\editionname

```
\providecommand*{\editionname}{edition}
```

\techreportname

```
\providecommand*{\techreportname}{Technical report}
```

\mscthesisname

```
\providecommand*{\mscthesisname}{Master's thesis}
```

\phdthesisname

```
\providecommand*{\phdthesisname}{PhD thesis}
```

6.5 Displaying the bibliography

```
\DTLbibliography{⟨bib dbname⟩}
```

Displays the bibliography for the database ⟨*bib dbname*⟩ which must have previously been loaded using `\DTLloadbbl`.

DTLbibliography

```
\newcommand*{\DTLbibliography}[2][\boolean{true}]{%
  \begin{DTLthebibliography}[#1]{#2}%
  \DTLforeachbibentry[#1]{#2}{%
    \DTLbibitem \DTLformatbibentry \DTLendbibitem
  }%
  \end{DTLthebibliography}%
}
```

TLformatbibentry

```
\DTLformatbibentry
```

Formats the current bib entry.

```
\newcommand*{\DTLformatbibentry}{%
```

Check format for this type is defined.

```
  \@ifundefined{DTLformat\DBIBentrytype}%
  {%
    \PackageError{databib}{Don't know how to format bibliography
    entries of type `\DBIBentrytype'}{}%
  }%
  {%
```

Print information to terminal and log file if in verbose mode.

```
    \dtl@message{[\DBIBcitekey]}%
```

Initialise

```
    \DTLstartsentencefalse\DTLmidsentencefalse\DTLperiodfalse
```

Format this entry

```
     \csname DTLformat\DBIBentrytype\endcsname
  }%
}
```

TLformatbibentry

```
\gDTLformatbibentry
```

Global version.

```
\newcommand*{\gDTLformatbibentry}{%
```

Check format for this type is defined.

```
  \@ifundefined{DTLformat\DBIBentrytype}%
  {%
    \PackageError{databib}{Don't know how to format bibliography
    entries of type `\DBIBentrytype'}{}%
  }%
  {%
```

Print information to terminal and log file if in verbose mode.

```
    \dtl@message{[\DBIBcitekey]}%
```

Initialise

```
    \global\DTLstartsentencefalse
    \global\DTLmidsentencefalse
    \global\DTLperiodfalse
```

Format this entry

```
     \csname DTLformat\DBIBentrytype\endcsname
  }%
}
```

rmatthisbibentry

```
\DTLformatthisbibentry{⟨db⟩}{⟨cite key⟩}
```

Just does \DTLformatbibentry for a given entry.

```
\newcommand*{\DTLformatthisbibentry}[2]{%
  \edef\DBIBname{#1}%
  \edef\DBIBcitekey{#2}%
  \edtlgetrowforvalue{#1}{\dtlcolumnindex{#1}{CiteKey}}{\DBIBcitekey}%
  \dtl@gathervalues{#1}{\dtlcurrentrow}%
  \letcs{\DBIBentrytype}{@dtl@key@EntryType}%
  \DTLformatbibentry
}
```

\DTLendbibitem Hook to add extra information at the end of a bibliography item. This does nothing by default.

```
\newcommand*{\DTLendbibitem}{}
```

\DTLwidest Define a length to store the widest bib entry label

```
\newlength\dtl@widest
```

tewidestbibentry

```
\DTLcomputewidestbibentry{⟨conditions⟩}{⟨db name⟩}{⟨bib label⟩}{⟨cmd⟩}
```

Computes the widest bibliography entry over all entries satisfying ⟨*condition*⟩ for the database called ⟨*db name*⟩, where the bibliography label is formated according to ⟨*bib label*⟩ and stores the result in ⟨*cmd*⟩ which must be a command name.

```
\newcommand*{\DTLcomputewidestbibentry}[4]{%
\dtl@widest=0pt\relax
\let#4=\@empty
\DTLforeachbibentry[#1]{#2}{%
\settowidth{\dtl@tmplength}{#3}%
\ifdim\dtl@tmplength>\dtl@widest\relax
 \dtl@widest=\dtl@tmplength
 \protected@edef#4{#3}%
\fi
}%
}
```

Lforeachbibentry

```
\DTLforeachbibentry[⟨condition⟩]{⟨db name⟩}{⟨text⟩}
```

```
\DTLforeachbibentry*[⟨condition⟩]{⟨db name⟩}{⟨text⟩}
```

Iterates through the database called ⟨*db name*⟩ and does ⟨*text*⟩ if ⟨*condition*⟩ is met. As with `\DTLforeach`, the starred version is read only.

```
\newcommand*{\DTLforeachbibentry}{%
\@ifstar\@sDTLforeachbibentry\@DTLforeachbibentry}
```

foreachbibentry

Unstarred version

```
\newcommand*{\@DTLforeachbibentry}[3][\boolean{true}]{%
```

Store database name.

```
  \edef\DBIBname{#2}%
```

Reset row counter.

```
  \setcounter{DTLbibrow}{0}%
```

Iterate through the database.

```
  \@DTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}{\refstepcounter{DTLbibrow}#3}{}%
  }%
}
```

foreachbibentry

Starred version

```
\newcommand*{\@sDTLforeachbibentry}[3][\boolean{true}]{%
```

Store database name.

```
  \edef\DBIBname{#2}%
```

Reset row counter.

```
  \setcounter{DTLbibrow}{0}%
```

Iterate through the database (read only).

```
  \@sDTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}{\refstepcounter{DTLbibrow}#3}{}%
  }%
}
```

Lforeachbibentry

```
\gDTLforeachbibentry[⟨condition⟩]{⟨db name⟩}{⟨text⟩}
```

```
\gDTLforeachbibentry*[⟨condition⟩]{⟨db name⟩}{⟨text⟩}
```

Global version.

```
\newcommand{\gDTLforeachbibentry}{%
\@ifstar\@sgDTLforeachbibentry\@gDTLforeachbibentry}
```

foreachbibentry

Unstarred version

```
\newcommand*{\@gDTLforeachbibentry}[3][\boolean{true}]{%
```

Store database name.

```
  \xdef\DBIBname{#2}%
```

Reset row counter.

```
  \global\c@DTLbibrow = 0\relax
```

Iterate through the database.

```
  \@DTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@g@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}%
    {%
      \refstepcounter{DTLbibrow}%
      \global\c@DTLbibrow=\c@DTLbibrow
      #3%
    }%
    {}%
  }%
}
```

foreachbibentry Starred version

```
\newcommand*{\@sgDTLforeachbibentry}[3][\boolean{true}]{%
```

Store database name.

```
  \xdef\DBIBname{#2}%
```

Reset row counter.

```
  \global\c@DTLbibrow = 0\relax
```

Iterate through the database (read only).

```
  \@sDTLforeach{#2}{\DBIBcitekey=CiteKey,\DBIBentrytype=EntryType}%
  {%
    \dtl@g@gathervalues{#2}{\dtlcurrentrow}%
    \ifthenelse{#1}%
    {%
      \refstepcounter{DTLbibrow}%
      \global\c@DTLbibrow=\c@DTLbibrow
      #3%
    }%
    {}%
  }%
}
```

DTLbibrow The counter DTLbibrow keeps track of the current row in the body of \DTLforeachbibentry. (You can't rely on DTLrowi, DTLrowii and DTLrowiii, as \DTLforeachbibentry pass the conditions to the optional argument of \DTLforeach, but instead uses \ifthenelse, which means that DTLrowi etc will be incremented, even when the given condition is not met.)

```
\newcounter{DTLbibrow}
```

\theHDTLbibrow Keep hyperref happy:

```
\def\theHDTLbibrow{\theHDTLrow.bib.\arabic{DTLbibrow}}%
```

\DTLbibfield

`\DTLbibfield{⟨field name⟩}`

Gets the value assigned to the field ⟨*field name*⟩ for the current row of \DTLforeachbibentry. (Doesn't check if the field exists, or if it is being used within \DTLforeachbibentry.)

```
\newcommand*{\DTLbibfield}[1]{\csname @dtl@key@#1\endcsname}
```

\DTLbibfieldlet

`\DTLbibfield{⟨cs⟩}{⟨field name⟩}`

Gets the value assigned to the field ⟨*field name*⟩ for the current row of \DTLforeachbibentry and assigns it to the control sequence ⟨*cs*⟩. (Doesn't check if the field exists, or if it is being used within \DTLforeachbibentry.)

```
\newcommand*{\DTLbibfieldlet}[2]{%
  \letcs{#1}{@dtl@key@#2}%
}
```

ifbibfieldexists

\DTLifbibfieldexists{⟨*field name*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines whether the given field name exists for the current row of `\DTLforeachbibentry`.

```
\newcommand*{\DTLifbibfieldexists}[3]{%
\@ifundefined{@dtl@key@#1}{#3}{%
\expandafter\DTLifnull\csname @dtl@key@#1\endcsname
{#3}{#2}}}
```

nybibfieldexists

\DTLifanybibfieldexists{⟨*list of field name*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

Determines whether any of the listed fields exist for the current row of `\DTLforeachbibentry`.

```
\newcommand*{\DTLifanybibfieldexists}[3]{%
\@for\dtl@thisfield:=#1\do{%
\@ifundefined{@dtl@key@\dtl@thisfield}{}{%
\expandafter\DTLifnull\csname @dtl@key@\dtl@thisfield\endcsname
{}{%
\@endfortrue}}}%
\if@endfor
 #2%
\else
 #3%
\fi
\@endforfalse
}
```

\ifDTLperiod The conditional `\ifDTLperiod` is used to keep track of any abbreviations ending with a period, this is to ensure that abbreviations aren't followed by a full stop if they already have a full stop terminating the abbreviation.

```
\newif\ifDTLperiod
```

Lcheckendsperiod

\DTLcheckendperiod{⟨*string*⟩}

Checks if ⟨*string*⟩ ends with a full stop. This sets `\ifDTLperiod`.

```
\newcommand*{\DTLcheckendsperiod}[1]{%
\dtl@checkendsperiod#1\@nil\relax}
```

```
\def\dtl@checkendsperiod#1#2{%
\def\@dtl@argi{#1}\def\@dtl@argii{#2}%
\def\@dtl@period{.}%
\ifx\@dtl@argi\@nnil
  \global\DTLperiodfalse
  \let\@dtl@donext=\relax
\else
  \ifx\@dtl@argii\@nnil
    \ifx\@dtl@argi\@dtl@period
      \global\DTLperiodtrue
    \else
      \global\DTLperiodfalse
    \fi
    \let\@dtl@donext=\@gobble
  \else
    \let\@dtl@donext=\dtl@checkendsperiod
  \fi
\fi
\@dtl@donext{#2}%
}
```

bfieldendsperiod

`\DTLcheckbibfieldendperiod{⟨label⟩}`

Checks if the bib field ⟨*label*⟩ ends with a full stop. This sets `\ifDTLperiod`.

```
\newcommand*{\DTLcheckbibfieldendsperiod}[1]{%
\protected@edef\@dtl@tmp{\DTLbibfield{#1}}%
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}}
```

ifDTLmidsentence

`\ifDTLmidsentence`

Determine whether we are in the middle of a sentence.

```
\newif\ifDTLmidsentence
```

DTLstartsentence

`\ifDTLstartsentence`

Determine whether we are at the start of a sentence.

```
\newif\ifDTLstartsentence
```

\DTLaddperiod

`\DTLaddperiod`

Adds a full stop and sets \DTLmidsentencefalse, \DTLstartsentencetrue and \DTLperiodfalse.

```
\newcommand*{\DTLaddperiod}{\DTLmidsentencefalse\DTLperiodfalse
\DTLstartsentencetrue
\ifDTLperiod\else.\fi}
```

\DTLaddcomma

\DTLaddcomma

Adds a comma and sets \DTLmidsentencetrue, \DTLperiodfalse and \DTLstartsentencefalse

```
\newcommand*{\DTLaddcomma}{, \DTLmidsentencetrue
\DTLperiodfalse\DTLstartsentencefalse}
```

rtsentencespace

Adds a space if at the start of the sentence, otherwise does nothing. (The space between sentences is added this way, rather than in \DTLaddperiod otherwise spurious extra space can occur at the end of the bib item. The spacefactor needs to be set manually, because there's stuff in the way of the previous sentence's full stop and this space which confuses the inter sentence spacing (and, of course, the previous sentence could have ended with a capital letter.)

```
\newcommand*{\DTLstartsentencespace}{%
\ifDTLstartsentence\spacefactor=\sfcode`\.\relax\space
\fi\DTLstartsentencefalse}
```

\DTLtwoand

In a list of only two author (or editor) names, the text between the two names is given by \DTLtwoand:

```
\newcommand*{\DTLtwoand}{\ \andname\ }
```

\DTLandlast

In a list of author (or editor) names, the text between the penultimate and last name is given by \DTLandlast:

```
\newcommand*{\DTLandlast}{, \andname\ }
```

\DTLandnotlast

In a list of author (or editor) names, the text between the names (except the penultimate and last name) is given by \DTLandnotlast:

```
\newcommand*{\DTLandnotlast}{, }
```

dtl@authorcount

Define a count register to keep track of the number of authors:

```
\newcount\@dtl@authorcount
```

DTLmaxauthors

The counter DTLmaxauthors indicates the maximum number of author names to display, if there are more than that number, \etalname is used.

```
\newcounter{DTLmaxauthors}
\setcounter{DTLmaxauthors}{10}
```

ormatauthorlist

Format a list of author names (the list is stored in \@dtl@key@Author):

```
\newcommand*{\DTLformatauthorlist}{%
\DTLifbibfieldexists{Author}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Author\do{%
\advance\@dtl@authorcount by 1\relax}%
\@dtl@tmpcount=0\relax
\ifnum\@dtl@authorcount>\c@DTLmaxauthors
{%
  \@for\@dtl@author:=\@dtl@key@Author\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformatauthor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount>\c@DTLmaxauthors
      \DTLandnotlast \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}%
      \@endfortrue
    \else
      \DTLandnotlast \expandafter\DTLformatauthor\@dtl@author
    \fi
  \fi
  }%
}%
\else
  \@for\@dtl@author:=\@dtl@key@Author\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformatauthor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount=\@dtl@authorcount
      \ifnum\@dtl@authorcount=2\relax
        \DTLtwoand
      \else
        \DTLandlast
      \fi
      \expandafter\DTLformatauthor\@dtl@author
    \else
      \DTLandnotlast \expandafter\DTLformatauthor\@dtl@author
    \fi
  \fi
  }%
\fi
}{}%
}
```

DTLmaxeditors The counter DTLmaxeditors indicates the maximum number of editor names to display, if there are more than that number, \etalname is used.

```
\newcounter{DTLmaxeditors}
\setcounter{DTLmaxeditors}{10}
```

ormateditorlist Format a list of editor names (the list is stored in \@dtl@key@Editor):

```
\newcommand*{\DTLformateditorlist}{%
\DTLifbibfieldexists{Editor}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\advance\@dtl@authorcount by 1\relax}%
\@dtl@tmpcount=0\relax
\ifnum\@dtl@authorcount>\c@DTLmaxeditors
{%
  \@for\@dtl@author:=\@dtl@key@Editor\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformateditor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount>\c@DTLmaxeditors
      \DTLandnotlast \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}%
      \@endfortrue
    \else
      \DTLandnotlast \expandafter\DTLformateditor\@dtl@author
    \fi
  \fi
  }%
}%
\else
\@for\@dtl@author:=\@dtl@key@Editor\do{%
  \advance\@dtl@tmpcount by 1\relax
  \ifnum\@dtl@tmpcount=1\relax
   \expandafter\DTLformateditor\@dtl@author
  \else
    \ifnum\@dtl@tmpcount=\@dtl@authorcount
      \ifnum\@dtl@authorcount=2\relax
        \DTLtwoand
      \else
        \DTLandlast
      \fi
      \expandafter\DTLformateditor\@dtl@author
    \else
      \DTLandnotlast \expandafter\DTLformateditor\@dtl@author
    \fi
  \fi
  }%
\fi
,
\ifnum\@dtl@authorcount=1\relax
  \editorname
  \expandafter\DTLcheckendsperiod\expandafter{\editorname}%
\else
```

```
  \editorsname
  \expandafter\DTLcheckendsperiod\expandafter{\editorsname}%
\fi
}{}%
}
```

ormatsurnameonly

\DTLformatsurnameonly{⟨*von part*⟩}{⟨*surname*⟩}{⟨*jr part*⟩}{⟨*forenames*⟩}

This is used when only the surname should be displayed. (The final argument, ⟨*forenames*⟩, is ignored.)

```
\newcommand*{\DTLformatsurnameonly}[4]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty\else#1~\fi
#2%
\def\@dtl@tmp{#3}%
\ifx\@dtl@tmp\@empty
  \DTLcheckendsperiod{#2}%
\else
  , #3%
  \DTLcheckendsperiod{#3}%
\fi
}
```

Lformatforenames

\DTLformatforenames{⟨*forenames*⟩}

The format of an author/editor's forenames. If the forenames occur at the start of sentence, a new sentence space is added. The argument is checked to determine whether it ends with a full stop (sometimes the forenames may include initials.)

```
\newcommand*{\DTLformatforenames}[1]{%
\DTLstartsentencespace
#1%
\DTLcheckendsperiod{#1}}
```

atabbrvforenames

\DTLformatabbrvforenames{⟨*forenames*⟩}

The format of an author/editor's abbreviated forenames. The initials may or may not end in a full stop depending on the commands governing the format of `\DTLstoreinitials`, so the initials need to be check using `\DTLcheckendsperiod`.

```
\newcommand*{\DTLformatabbrvforenames}[1]{%
\DTLstartsentencespace
\DTLstoreinitials{#1}{\@dtl@tmp}\@dtl@tmp
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}}
```

\DTLformatvon

`\DTLformatvon{⟨von part⟩}`

The format of the "von" part. This does nothing if the argument is empty, otherwise it does the argument followed by a non-breakable space.

```
\newcommand*{\DTLformatvon}[1]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty
\else
  #1~%
\fi
}
```

DTLformatsurname

`\DTLformatsurname{⟨surname⟩}`

The format of an author/editor's surname.

```
\newcommand*{\DTLformatsurname}[1]{%
\DTLstartsentencespace
#1\DTLcheckendsperiod{#1}}
```

\DTLformatjr

`\DTLformatjr{⟨jr part⟩}`

The format of the "jr" part. This does nothing if the argument is empty.

```
\newcommand*{\DTLformatjr}[1]{%
\DTLstartsentencespace
\def\@dtl@tmp{#1}%
\ifx\@dtl@tmp\@empty
\else
  , #1\DTLcheckendsperiod{#1}%
\fi
}
```

tcrossrefeditor

Format cross reference editors:

```
\newcommand*{\DTLformatcrossrefeditor}{%
\DTLifbibfieldexists{Editor}{%
\DTLstartsentencespace
\@dtl@authorcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\advance\@dtl@authorcount by 1\relax}%
{\@dtl@tmpcount=0\relax
\@for\@dtl@author:=\@dtl@key@Editor\do{%
\ifnum\@dtl@authorcount=1\relax
 \expandafter\DTLformatsurnameonly\@dtl@author
\else
 \advance\@dtl@tmpcount by 1\relax
 \ifnum\@dtl@tmpcount=1\relax
    \expandafter\DTLformatsurnameonly\@dtl@author
  \else
    \ifnum\@dtl@authorcount=2\relax
      \ \andname\ \expandafter\DTLformatsurnameonly\@dtl@author
    \else
      \ \etalname
      \expandafter\DTLcheckendsperiod\expandafter{\etalname}
    \fi
    \@endfortrue
  \fi
\fi
}}%
}{}%
}
```

rmatvolnumpages Format volume, number and pages (of an article).

```
\newcommand*{\DTLformatvolnumpages}{%
\DTLifbibfieldexists{Volume}{%
\DTLstartsentencespace
\DTLbibfield{Volume}\DTLperiodfalse}{}%
\DTLifbibfieldexists{Number}{%
\DTLstartsentencespace
(\DTLbibfield{Number})\DTLperiodfalse}{}%
\DTLifbibfieldexists{Pages}{%
\DTLifanybibfieldexists{Volume,Number}{:}{%
\DTLstartsentencespace
\protected@edef\@dtl@pages{0\DTLbibfield{Pages}}%
\DTLifnumerical{\@dtl@pages}{\pagename}{\pagesname}~}%
\DTLbibfield{Pages}\DTLperiodfalse}{}%
}
```

TLformatbvolume Format book volume.

```
\newcommand*{\DTLformatbvolume}{%
\DTLifbibfieldexists{Volume}{%
\ifDTLmidsentence
  \volumename
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\volumename
\fi
~\DTLbibfield{Volume}%
\DTLifbibfieldexists{Series}{\ \ofname\
{\em\DTLbibfield{Series}}\DTLcheckbibfieldendsperiod{Series}}{%
\DTLcheckbibfieldendsperiod{Volume}}%
}{}}
```

matchapterpages Format chapter and pages:

```
\newcommand*{\DTLformatchapterpages}{%
\DTLifbibfieldexists{Chapter}{%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}}{%
\DTLstartsentencespace
\chaptername}~\DTLbibfield{Chapter}%
\DTLifbibfieldexists{Pages}{\DTLaddcomma}{%
\DTLcheckbibfieldendsperiod{Chapter}}}{}%
\DTLstartsentencespace
\DTLformatpages}
```

\DTLformatpages Format pages:

```
\newcommand*{\DTLformatpages}{%
\DTLifbibfieldexists{Pages}{%
\DTLstartsentencespace
\protected@edef\@dtl@pages{0\DTLbibfield{Pages}}%
\DTLifnumerical{\@dtl@pages}{\pagename}{\pagesname}~%
\DTLbibfield{Pages}\DTLcheckbibfieldendsperiod{Pages}}{}%
}
```

matnumberseries Format number and series (of book)

```
\newcommand*{\DTLformatnumberseries}{%
\DTLifbibfieldexists{Volume}{}{%
\DTLifbibfieldexists{Number}{%
\ifDTLmidsentence
  \numbername
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\numbername
\fi~\DTLbibfield{Number}%
\DTLifbibfieldexists{Series}{\ \inname\ \DTLbibfield{Series}%
\DTLcheckbibfieldendsperiod{Series}}{%
\DTLcheckbibfieldendsperiod{Number}}%
}{%
\DTLifbibfieldexists{Series}{%
\DTLstartsentencespace
\DTLbibfield{Series}%
\DTLcheckbibfieldendsperiod{Series}}{}}%
}%
}
```

matbookcrossref Format a book cross reference.

```
\newcommand*{\DTLformatbookcrossref}{%
\DTLifbibfieldexists{Volume}{%
\ifDTLmidsentence
  \volumename
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\volumename
\fi
~\DTLbibfield{Volume}\ \ofname\
}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ }%
\DTLifbibfieldexists{Editor}{\DTLformatcrossrefeditor}{%
\DTLifbibfieldexists{Key}{%
\DTLbibfield{Key}}{%
\DTLifbibfieldexists{Series}{%
{\em\DTLbibfield{Series}}}{}%
}%
}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
```

ollproccrossref Format ‘incollections’ cross reference.

```
\newcommand*{\DTLformatincollproccrossref}{%
\DTLifbibfieldexists{Editor}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\
\DTLformatcrossrefeditor
}{%
\DTLifbibfieldexists{Key}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ \DTLbibfield{Key}%
}{%
\DTLifbibfieldexists{BookTitle}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\ \DTLformatbooktitle{\DTLbibfield{BookTitle}}}{}%
}}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
```

atinedbooktitle Format editor and booktitle:

```
\newcommand*{\DTLformatinedbooktitle}{%
\DTLifbibfieldexists{BookTitle}{%
\ifDTLmidsentence
  \inname
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\inname
\fi\
\DTLifbibfieldexists{Editor}{%
\DTLformateditorlist\DTLaddcomma \DTLformatbooktitle{\DTLbibfield{BookTitle}}%
\DTLcheckbibfieldendsperiod{BookTitle}%
}{\DTLformatbooktitle{\DTLbibfield{BookTitle}}%
\DTLcheckbibfieldendsperiod{BookTitle}%
}}{}}
```

\DTLformatdate Format date.

```
\newcommand*{\DTLformatdate}{%
\DTLifbibfieldexists{Year}{%
\DTLifbibfieldexists{Month}{%
\protected@edef\@dtl@tmp{\DTLbibfield{Month}}%
\ifDTLmidsentence
  \@dtl@tmp
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\@dtl@tmp
\fi\
\DTLmidsentencefalse}{}%
\DTLstartsentencespace
\DTLbibfield{Year}}{%
\DTLifbibfieldexists{Month}{%
\protected@edef\@dtl@tmp{\DTLbibfield{Month}}%
\ifDTLmidsentence
  \@dtl@tmp
\else
  \DTLstartsentencespace
  \expandafter\MakeUppercase\@dtl@tmp
\fi
\DTLcheckbibfieldendsperiod{Month}%
}{}}}
```

articlecrossref Format article cross reference.

```
\newcommand*{\DTLformatarticlecrossref}{%
\DTLifbibfieldexists{Key}{%
\ifDTLmidsentence
 \inname
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\inname
\fi
\ {\em\DTLbibfield{Key}}}{%
\DTLifbibfieldexists{Journal}{%
\ifDTLmidsentence
 \inname
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\inname
\fi
\ {\em\DTLbibfield{Journal}}}{}}%
~\DTLpcite{\DTLbibfield{CrossRef}}%
}
```

\DTLpcite

```
\newrobustcmd*{\DTLpcite}[1]{%
  \protected@edef\@dtl@tmp{#1}%
  \cite{\@dtl@tmp}%
}
```

6.5.1 ifthen conditionals

The conditionals defined in this section may be used in the optional argument of `\DTLforeachbibentry`. They may also be used in the first argument of `\ifthenelse`, but only if the command occurs within the body of `\DTLforeachbibentry`.

TLbibfieldexists

`\DTLbibfieldexists{⟨field label⟩}`

Checks if named bib field exists for current entry

```
\newcommand*{\DTLbibfieldexists}[1]{%
\TE@throw\noexpand\dtl@testbibfieldexists{#1}%
\noexpand\if@dtl@condition}
```

tbibfieldexists

```
\newcommand*{\dtl@testbibfieldexists}[1]{%
\DTLifbibfieldexists{#1}{\@dtl@conditiontrue}{\@dtl@conditionfalse}}
```

\DTLbibfieldiseq

\DTLbibfieldiseq{⟨*field label*⟩}{⟨*value*⟩}

Checks if the value of the bib field given by ⟨*field label*⟩ is equal to ⟨*value*⟩. (Uses `\dtlcompare` to determine if the values are equal. If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldiseq}[2]{%
\TE@throw\noexpand\dtl@testbibfieldiseq{#1}{#2}%
\noexpand\if@dtl@condition}
```

estbibfieldiseq

```
\newcommand*{\dtl@testbibfieldiseq}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=0\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
```

\DTLbibfieldislt

\DTLbibfieldislt{⟨*field label*⟩}{⟨*value*⟩}

Checks if the value of the bib field given by ⟨*field label*⟩ is less than ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldislt}[2]{%
\TE@throw\noexpand\dtl@testbibfieldislt{#1}{#2}%
\noexpand\if@dtl@condition}
```

estbibfieldislt

```
\newcommand*{\dtl@testbibfieldislt}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=-1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
```

\DTLbibfieldisle

`\DTLbibfieldisle{⟨field label⟩}{⟨value⟩}`

Checks if the value of the bib field given by ⟨*field label*⟩ is less than or equal to ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisle}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisle{#1}{#2}%
\noexpand\if@dtl@condition}
```

estbibfieldisle

```
\newcommand*{\dtl@testbibfieldisle}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount<1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
```

\DTLbibfieldisgt

`\DTLbibfieldisgt{⟨field label⟩}{⟨value⟩}`

Checks if the value of the bib field given by ⟨*field label*⟩ is greater than ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisgt}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisgt{#1}{#2}%
\noexpand\if@dtl@condition}
```

estbibfieldisgt

```
\newcommand*{\dtl@testbibfieldisgt}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount=1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
```

\DTLbibfieldisge

`\DTLbibfieldisge{⟨field label⟩}{⟨value⟩}`

Checks if the value of the bib field given by ⟨*field label*⟩ is less than or equal to ⟨*value*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldisge}[2]{%
\TE@throw\noexpand\dtl@testbibfieldisge{#1}{#2}%
\noexpand\if@dtl@condition}
```

estbibfieldisge

```
\newcommand*{\dtl@testbibfieldisge}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
 =\csname @dtl@key@#1\endcsname
\expandafter\toks@\expandafter{\@dtl@tmp}%
\@dtl@toks{#2}%
\edef\@dtl@docompare{\noexpand\dtlcompare{\noexpand\@dtl@tmpcount}%
{\the\toks@}{\the\@dtl@toks}}%
\@dtl@docompare
\ifnum\@dtl@tmpcount>-1\relax
 \@dtl@conditiontrue
\else
 \@dtl@conditionfalse
\fi
}{%
\@dtl@conditionfalse}%
}
```

bibfieldcontains

```
\DTLbibfieldcontains{⟨field label⟩}{⟨sub string⟩}
```

Checks if the value of the bib field given by ⟨*field label*⟩ contains ⟨*sub string*⟩. (If the bib field doesn't exist, the condition is false.)

```
\newcommand*{\DTLbibfieldcontains}[2]{%
\TE@throw\noexpand\dtl@testbibfieldcontains{#1}{#2}%
\noexpand\if@dtl@condition}
```

ibfieldcontains

```
\newcommand*{\dtl@testbibfieldcontains}[2]{%
\DTLifbibfieldexists{#1}{%
\expandafter\let\expandafter\@dtl@tmp\expandafter
  =\csname @dtl@key@#1\endcsname
\expandafter\dtl@testifsubstring\expandafter{\@dtl@tmp}{#2}%
}{\@dtl@conditionfalse}}
```

6.6 Bibliography Style Macros

The macros defined in this section should be redefined by bibliography styles.

thebibliography How to format the entire bibliography:

```
\newenvironment{DTLthebibliography}[2][\boolean{true}]{%
\@dtl@tmpcount=0\relax
\@sDTLforeach[#1]{#2}{}{\advance\@dtl@tmpcount by 1\relax}%
\begin{thebibliography}{\number\@dtl@tmpcount}
}{\end{thebibliography}}
```

\DTLmonthname The monthname style. The argument must be a number from 1 to 12. By default, uses `\dtl@monthname`.

```
\newcommand*{\DTLmonthname}[1]{%
\dtl@monthname{#1}}
```

\dtl@monthname Full month names:

```
\newcommand*{\dtl@monthname}[1]{%
\ifcase#1%
\or January%
\or February%
\or March%
\or April%
\or May%
\or June%
\or July%
\or August%
\or September%
\or October%
\or November%
\or December%
\fi}
```

@abbrvmonthname

Abbreviated months:

```
\newcommand*{\dtl@abbrvmonthname}[1]{%
\ifcase#1%
\or Jan.%
\or Feb.%
\or Mar.%
\or Apr.%
\or May%
\or June%
\or July%
\or Aug.%
\or Sept.%
\or Oct.%
\or Nov.%
\or Dec.%
\fi}
```

\DTLbibitem

Define how to start a new bibitem:

```
\newcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}
```

\DTLmbibitem

As `\DTLbibitem` but for `\DTLmbibliography`

```
\newcommand*{\DTLmbibitem}[1]{\bibitem{#1@\DBIBcitekey}}
```

DTLcustombibitem

`\DTLcustombibitem{⟨item code⟩}{⟨ref text⟩}{⟨cite key⟩}`

As `\DTLbibitem` but user provides ⟨*item code*⟩ to use in place of `\item`. This code can access the cite key using `\DBIBcitekey`. The ⟨*ref text*⟩ is the text associated with this bib item. (For example, if used in an enumerate environment, ⟨*ref text*⟩ might be `\theenumi`.)

```
\newcommand*{\DTLcustombibitem}[3]{%
  #1%
  \if@filesw
    \immediate\write\@auxout{\string\bibcite{#3}{#2}}%
  \fi
  \ignorespaces
}
```

\DTLformatauthor

`\DTLformatauthor{⟨von part⟩}{⟨surname⟩}{⟨junior part⟩}{⟨forenames⟩}`

The format of an author's name.

```
\newcommand*{\DTLformatauthor}[4]{%
\DTLformatforenames{#4}
\DTLformatvon{#1}%
\DTLformatsurname{#2}%
\DTLformatjr{#3}}
```

DTLformateditor The format of an editor's name.

```
\newcommand*{\DTLformateditor}[4]{%
\DTLformatforenames{#4}
\DTLformatvon{#1}%
\DTLformatsurname{#2}%
\DTLformatjr{#3}}
```

TLformatedition The format of an edition:

```
\newcommand*{\DTLformatedition}[1]{#1 \editionname}
```

TLformatarticle The format of an article:

```
\newcommand{\DTLformatarticle}{}
```

\DTLformatbook The format of a book:

```
\newcommand{\DTLformatbook}{}
```

TLformatbooklet The format of a booklet:

```
\newcommand{\DTLformatbooklet}{}
```

DTLformatinbook The format of an "inbook" type:

```
\newcommand{\DTLformatinbook}{}
```

matincollection The format of an "incollection" type:

```
\newcommand{\DTLformatincollection}{}
```

atinproceedings The format of an "inproceedings" type:

```
\newcommand{\DTLformatinproceedings}{}
```

DTLformatmanual The format of a manual:

```
\newcommand{\DTLformatmanual}{}
```

atmastersthesis The format of a master's thesis:

```
\newcommand{\DTLformatmastersthesis}{}
```

\DTLformatmisc The format of a miscellaneous entry:

```
\newcommand{\DTLformatmisc}{}
```

formatphdthesis The format of a Ph.D. thesis:

```
\newcommand{\DTLformatphdthesis}{}
```

rmatproceedings The format of a proceedings:

```
\newcommand{\DTLformatproceedings}{}
```

ormattechreport

The format of a technical report:

```
\newcommand{\DTLformattechreport}{}
```

rmatunpublished

The format of an unpublished work:

```
\newcommand{\DTLformatunpublished}{}
```

Predefined names (these correspond to the standard BibTeX predefined strings of the same name without the leading \DTL):

\DTLacmcs

```
\newcommand*{\DTLacmcs}{ACM Computing Surveys}
```

\DTLacta

```
\newcommand*{\DTLacta}{Acta Informatica}
```

\DTLcacm

```
\newcommand*{\DTLcacm}{Communications of the ACM}
```

\DTLibmjrd

```
\newcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
```

\DTLibmsj

```
\newcommand*{\DTLibmsj}{IBM Systems Journal}
```

\DTLieeese

```
\newcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
```

\DTLieeetc

```
\newcommand*{\DTLieeetc}{IEEE Transactions on Computers}
```

\DTLieeetcad

```
\newcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
of Integrated Circuits}
```

\DTLipl

```
\newcommand*{\DTLipl}{Information Processing Letters}
```

\DTLjacm

```
\newcommand*{\DTLjacm}{Journal of the ACM}
```

\DTLjcss

```
\newcommand*{\DTLjcss}{Journal of Computer and System Sciences}
```

\DTLscp

```
\newcommand*{\DTLscp}{Science of Computer Programming}
```

\DTLsicomp

```
\newcommand*{\DTLsicomp}{SIAM Journal on Computing}
```

\DTLtocs

```
\newcommand*{\DTLtocs}{ACM Transactions on Computer Systems}
```

\DTLtods

```
\newcommand*{\DTLtods}{ACM Transactions on Database Systems}
```

\DTLtog

```
\newcommand*{\DTLtog}{ACM Transactions on Graphics}
```

\DTLtoms

```
\newcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}
```

\DTLtoois

```
\newcommand*{\DTLtoois}{ACM Transactions on Office Information
Systems}
```

\DTLtoplas

```
\newcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
and Systems}
```

\DTLtcs

```
\newcommand*{\DTLtcs}{Theoretical Computer Science}
```

6.7 Bibliography Styles

Each bibliography style is set by the command \dtlbst@⟨*style*⟩, where ⟨*style*⟩ is the name of the bibliography style.

\dtlbst@plain The 'plain' style:

```
\newcommand{\dtlbst@plain}{%
```

Set how to format the entire bibliography:

```
  \renewenvironment{DTLthebibliography}[2][\boolean{true}]{%
  \@dtl@tmpcount=0\relax
  \@sDTLforeach[##1]{##2}{}{\advance\@dtl@tmpcount by 1\relax}%
  \begin{thebibliography}{\number\@dtl@tmpcount}%
  }{\end{thebibliography}}%
```

Set how to start the bibliography entry:

```
  \renewcommand*{\DTLbibitem}{\bibitem{\DBIBcitekey}}%
  \renewcommand*{\DTLmbibitem}[1]{\bibitem{##1@\DBIBcitekey}}%
```

Sets the author name format.

```
\renewcommand*{\DTLformatauthor}[4]{%
\DTLformatforenames{##4}
\DTLformatvon{##1}%
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
```

Sets the editor name format.

```
\renewcommand*{\DTLformateditor}[4]{%
\DTLformatforenames{##4}
\DTLformatvon{##1}%
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
```

Sets the edition format.

```
\renewcommand*{\DTLformatedition}[1]{##1 \editionname}%
```

Sets the monthname format.

```
\let\DTLmonthname\dtl@monthname
```

Sets other predefined names:

```
  \renewcommand*{\DTLacmcs}{ACM Computing Surveys}
  \renewcommand*{\DTLacta}{Acta Informatica}
  \renewcommand*{\DTLcacm}{Communications of the ACM}
  \renewcommand*{\DTLibmjrd}{IBM Journal of Research and Development}
  \renewcommand*{\DTLibmsj}{IBM Systems Journal}
  \renewcommand*{\DTLieeese}{IEEE Transactions on Software Engineering}
  \renewcommand*{\DTLieeetc}{IEEE Transactions on Computers}
  \renewcommand*{\DTLieeetcad}{IEEE Transactions on Computer-Aided Design
of Integrated Circuits}
  \renewcommand*{\DTLipl}{Information Processing Letters}
  \renewcommand*{\DTLjacm}{Journal of the ACM}
  \renewcommand*{\DTLjcss}{Journal of Computer and System Sciences}
  \renewcommand*{\DTLscp}{Science of Computer Programming}
  \renewcommand*{\DTLsicomp}{SIAM Journal on Computing}
  \renewcommand*{\DTLtocs}{ACM Transactions on Computer Systems}
  \renewcommand*{\DTLtods}{ACM Transactions on Database Systems}
  \renewcommand*{\DTLtog}{ACM Transactions on Graphics}
  \renewcommand*{\DTLtoms}{ACM Transactions on Mathematical Software}
  \renewcommand*{\DTLtoois}{ACM Transactions on Office Information
Systems}
  \renewcommand*{\DTLtoplas}{ACM Transactions on Programming Languages
and Systems}
  \renewcommand*{\DTLtcs}{Theoretical Computer Science}
```

The format of an article.

```
\renewcommand*{\DTLformatarticle}{%
  \DTLformatauthorlist
  \DTLifbibfieldexists{Author}{\DTLaddperiod}{}%
  \DTLifbibfieldexists{Title}{%
  \DTLstartsentencespace\DTLbibfield{Title}%
```

```
    \DTLcheckbibfieldendsperiod{Title}%
    \DTLaddperiod}{}%
    \DTLifbibfieldexists{CrossRef}{%
% cross ref field
    \DTLformatarticlecrossref
    \DTLifbibfieldexists{Pages}{\DTLaddcomma}{}%
    \DTLformatpages
    \DTLaddperiod
    }{% no cross ref field
    \DTLifbibfieldexists{Journal}{\DTLstartsentencespace
    {\em\DTLbibfield{Journal}}%
    \DTLcheckbibfieldendsperiod{Journal}%
    \DTLifanybibfieldexists{Number,Volume,Pages,Month,Year}{%
    \DTLaddcomma}{\DTLaddperiod}}{}%
    \DTLformatvolnumpages
    \DTLifanybibfieldexists{Volume,Number,Pages}{%
    \DTLifanybibfieldexists{Year,Month}{\DTLaddcomma}{%
    \DTLaddperiod}%
    \DTLmidsentencefalse}{}%
    \DTLformatdate
    \DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
    }%
    \DTLifbibfieldexists{Note}{\DTLstartsentencespace\DTLbibfield{Note}%
    \DTLcheckbibfieldendsperiod{Note}%
    \DTLaddperiod}{}%
  }
```

The format of a book.

```
\renewcommand*{\DTLformatbook}{%
  \DTLifbibfieldexists{Author}%
  {%
    \DTLformatauthorlist\DTLaddperiod
  }%
  {%
    \DTLformateditorlist
    \DTLifbibfieldexists{Editor}%
    {%
      \DTLaddperiod
    }%
    {}%
  }%
  \DTLifbibfieldexists{Title}%
  {%
    \DTLstartsentencespace
    \DTLformatbooktitle{\DTLbibfield{Title}}%
    \DTLcheckbibfieldendsperiod{Title}%
  }%
  {}%
  \DTLifbibfieldexists{CrossRef}%
  {%
```

Cross ref field

```
      \DTLifbibfieldexists{Title}{\DTLaddperiod}{}%
      \DTLformatbookcrossref
      \DTLifanybibfieldexists{Edition,Month,Year}%
      {\DTLaddcomma}%
      {\DTLaddperiod}%
    }%
    {%
```

no cross ref field

```
      \DTLifbibfieldexists{Title}%
      {%
        \DTLifbibfieldexists{Volume}{\DTLaddcomma}{\DTLaddperiod}%
      }%
      {}%
      \DTLformatbvolume
      \DTLformatnumberseries
      \DTLifanybibfieldexists{Number,Series,Volume}{\DTLaddperiod}{}%
      \DTLifbibfieldexists{Publisher}%
      {%
        \DTLstartsentencespace
        \DTLbibfield{Publisher}%
        \DTLcheckbibfieldendsperiod{Publisher}%
        \DTLifbibfieldexists{Address}%
        {\DTLaddcomma}%
        {%
          \DTLifanybibfieldexists{Month,Year}%
          {\DTLaddcomma}%
          {\DTLaddperiod}%
        }%
      }%
      {}%
      \DTLifbibfieldexists{Address}%
      {%
        \DTLstartsentencespace
        \DTLbibfield{Address}%
        \DTLcheckbibfieldendsperiod{Address}%
        \DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
      }%
      {}%
    }%
    \DTLifbibfieldexists{Edition}%
    {%
      \protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
      \ifDTLmidsentence
       \@dtl@tmp
      \else
       \DTLstartsentencespace\expandafter\MakeUppercase\@dtl@tmp
      \fi
      \expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
      \DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
    }%
    {}%
    \DTLformatdate
    \DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
    \DTLifbibfieldexists{Note}%
    {%
      \DTLstartsentencespace
      \DTLbibfield{Note}%
      \DTLcheckbibfieldendsperiod{Note}%
      \DTLaddperiod
    }%
    {}%
  }%
```

The format of a booklet.

```
\renewcommand*{\DTLformatbooklet}{%
\DTLifbibfieldexists{Author}{%
\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{HowPublished}{%
\DTLstartsentencespace\DTLbibfield{HowPublished}%
\DTLcheckbibfieldendsperiod{HowPublished}%
\DTLifanybibfieldexists{Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Year,Month}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{\DTLstartsentencespace\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of an ‘inbook’ entry.

```
\renewcommand*{\DTLformatinbook}{%
\DTLifbibfieldexists{Author}{%
\DTLformatauthorlist\DTLaddperiod}{%
\DTLifbibfieldexists{Editor}{\DTLformateditorlist\DTLaddperiod}{}}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
}{}%
\DTLifbibfieldexists{CrossRef}{%
% Cross ref entry
\DTLifbibfieldexists{Title}{%
\DTLifbibfieldexists{Chapter}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLformatbookcrossref
}{% no cross ref
\DTLifbibfieldexists{Title}{%
\DTLifanybibfieldexists{Chapter,Volume}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifanybibfieldexists{Volume,Series}{%
\DTLifanybibfieldexists{Chapter,Pages}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifbibfieldexists{Address}{\DTLaddcomma}{}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}}{}%
}%
\DTLifanybibfieldexists{Edition,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of an ‘incollection’ entry.

```
\renewcommand*{\DTLformatincollection}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{CrossRef}{%
% cross ref entry
\DTLformatincollproccrossref
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddcomma}{}%
\DTLformatchapterpages\DTLaddperiod
}{% no cross ref entry
\DTLformatinedbooktitle
\DTLifbibfieldexists{BookTitle}{%
\DTLifanybibfieldexists{Volume,Series,Chapter,Pages,Number}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Series,Chapter,Pages}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatnumberseries
\DTLifanybibfieldexists{Number,Series}{%
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatchapterpages
\DTLifanybibfieldexists{Chapter,Pages}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}%
}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of an 'inproceedings' entry.

```
\renewcommand*{\DTLformatinproceedings}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist
\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{CrossRef}{%
% cross ref entry
\DTLformatincollproccrossref
\DTLifbibfieldexists{Pages}{\DTLaddcomma}{%
\DTLaddperiod}%
\DTLformatpages
\DTLifbibfieldexists{Pages}{\DTLaddperiod}{}%
}{% no cross ref
\DTLformatinedbooktitle
\DTLifbibfieldexists{BookTitle}{%
\DTLifanybibfieldexists{Volume,Series,Pages,Number,Address,%
Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Series,Pages,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatnumberseries
\DTLifanybibfieldexists{Number,Series}{%
\DTLifanybibfieldexists{Pages,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatpages
\DTLifbibfieldexists{Pages}{%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Publisher}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLaddperiod}{}%
}{%
\DTLifanybibfieldexists{Publisher,Organization}{%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Publisher,Month,Year}{%
\DTLaddcomma}{}}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a manual.

```
\renewcommand*{\DTLformatmanual}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist
\DTLaddperiod}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Address}{\DTLaddcomma \DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
}{}%
\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifbibfieldexists{Author}{%
\DTLifanybibfieldexists{Organization,Address}{%
\DTLaddperiod}{\DTLaddcomma}}{%
\DTLifanybibfieldexists{Organization,Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}}{}%
\DTLifbibfieldexists{Author}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Address,Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
}{%
\DTLifbibfieldexists{Organization}{}{%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Edition,Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}}%
}%
\DTLifbibfieldexists{Edition}{%
\protected@edef\@dtl@tmp{\DTLformatedition{\DTLbibfield{Edition}}}%
\ifDTLmidsentence
 \@dtl@tmp
\else
 \DTLstartsentencespace
 \expandafter\MakeUppercase\@dtl@tmp
\fi
\expandafter\DTLcheckendsperiod\expandafter{\@dtl@tmp}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
```

```
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a master's thesis.

```
\renewcommand*{\DTLformatmastersthesis}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifanybibfieldexists{School,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{School}{%
\DTLstartsentencespace
\DTLbibfield{School}%
\DTLcheckbibfieldendsperiod{School}%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a miscellaneous entry.

```
\renewcommand*{\DTLformatmisc}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifbibfieldexists{HowPublished}{\DTLaddperiod}{%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}%
}%
\DTLmidsentencefalse}{}%
\DTLifbibfieldexists{HowPublished}{%
\DTLstartsentencespace
\DTLbibfield{HowPublished}%
\DTLcheckbibfieldendsperiod{HowPublished}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{%
\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a PhD thesis.

```
\renewcommand*{\DTLformatphdthesis}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifanybibfieldexists{School,Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{School}{%
\DTLstartsentencespace
\DTLbibfield{School}%
\DTLcheckbibfieldendsperiod{School}%
\DTLifanybibfieldexists{Address,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a proceedings.

```
\renewcommand*{\DTLformatproceedings}{%
\DTLifbibfieldexists{Editor}{%
\DTLformateditorlist\DTLaddperiod}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLaddperiod}{}}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
{\em\DTLbibfield{Title}}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLifanybibfieldexists{Volume,Number,Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
}{}%
\DTLformatbvolume
\DTLifbibfieldexists{Volume}{%
\DTLifanybibfieldexists{Number,Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatnumberseries
\DTLifbibfieldexists{Number}{%
\DTLifanybibfieldexists{Address,Editor,Publisher,%
Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Editor}{\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifbibfieldexists{Publisher}{%
\DTLaddcomma}{\DTLaddperiod}}{}}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLaddperiod
}{}%
}{% no address
\DTLifbibfieldexists{Editor}{%
\DTLifbibfieldexists{Organization}{%
\DTLstartsentencespace
\DTLbibfield{Organization}%
\DTLcheckbibfieldendsperiod{Organization}%
\DTLifanybibfieldexists{Publisher,Month,Year}{%
\DTLaddcomma}{\DTLaddperiod}}{}%
}{}%
\DTLifbibfieldexists{Publisher}{%
\DTLstartsentencespace
\DTLbibfield{Publisher}%
\DTLcheckbibfieldendsperiod{Publisher}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of a technical report.

```
\renewcommand*{\DTLformattechreport}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Type}{%
\DTLstartsentencespace
\DTLbibfield{Type}%
\DTLcheckbibfieldendsperiod{Type}%
\DTLifbibfieldexists{Number}{~}{}}{}%
\DTLifbibfieldexists{Number}{%
\DTLstartsentencespace
\DTLbibfield{Number}%
\DTLcheckbibfieldendsperiod{Number}%
}{}%
\DTLifanybibfieldexists{Type,Number}{%
\DTLifanybibfieldexists{Institution,Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Institution}{%
\DTLstartsentencespace
\DTLbibfield{Institution}%
\DTLcheckbibfieldendsperiod{Institution}%
\DTLifanybibfieldexists{Address,Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLifbibfieldexists{Address}{%
\DTLstartsentencespace
\DTLbibfield{Address}%
\DTLcheckbibfieldendsperiod{Address}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma
}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLaddperiod}{}%
}%
```

The format of an unpublished work.

```
\renewcommand*{\DTLformatunpublished}{%
\DTLifbibfieldexists{Author}{\DTLformatauthorlist\DTLaddperiod}{}%
\DTLifbibfieldexists{Title}{%
\DTLstartsentencespace
\DTLbibfield{Title}%
\DTLcheckbibfieldendsperiod{Title}%
\DTLaddperiod}{}%
\DTLifbibfieldexists{Note}{%
\DTLstartsentencespace
\DTLbibfield{Note}%
\DTLcheckbibfieldendsperiod{Note}%
\DTLifanybibfieldexists{Month,Year}{\DTLaddcomma}{\DTLaddperiod}}{}%
\DTLformatdate
\DTLifanybibfieldexists{Month,Year}{\DTLaddperiod}{}%
}%
```

End of 'plain' style.

```
}
```

formatbooktitle

```
\newcommand*{\DTLformatbooktitle}[1]{\emph{#1}}
```

\dtlbst@abbrv Define 'abbrv' style. This is similar to 'plain' except that some of the values are abbreviated

```
\newcommand{\dtlbst@abbrv}{%
```

Base this style on 'plain':

```
\dtlbst@plain
```

Sets the author name format.

```
\renewcommand*{\DTLformatauthor}[4]{%
\DTLformatabbrvforenames{##4}
\DTLformatvon{##1}%
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
```

Sets the editor name format.

```
\renewcommand*{\DTLformateditor}[4]{%
\DTLformatabbrvforenames{##4}
\DTLformatvon{##1}%
\DTLformatsurname{##2}%
\DTLformatjr{##3}}
```

Sets the monthname format.

```
\let\DTLmonthname\dtl@abbrvmonthname
```

Sets other predefined names:

```
\renewcommand*{\DTLacmcs}{ACM Comput.\ Surv.}
\renewcommand*{\DTLacta}{Acta Inf.}
\renewcommand*{\DTLcacm}{Commun.\ ACM}
\renewcommand*{\DTLibmjrd}{IBM J.\ Res.\ Dev.}
\renewcommand*{\DTLibmsj}{IBM Syst.~J.}
\renewcommand*{\DTLieeese}{IEEE Trans. Softw.\ Eng.}
\renewcommand*{\DTLieeetc}{IEEE Trans.\ Comput.}
\renewcommand*{\DTLieeetcad}{IEEE Trans.\ Comput.-Aided Design
Integrated Circuits}
\renewcommand*{\DTLipl}{Inf.\ Process.\ Lett.}
\renewcommand*{\DTLjacm}{J.~ACM}
\renewcommand*{\DTLjcss}{J.~Comput.\ Syst.\ Sci.}
\renewcommand*{\DTLscp}{Sci.\ Comput.\ Programming}
\renewcommand*{\DTLsicomp}{SIAM J.~Comput.}
\renewcommand*{\DTLtocs}{ACM Trans.\ Comput.\ Syst.}
\renewcommand*{\DTLtods}{ACM Trans.\ Database Syst.}
\renewcommand*{\DTLtog}{ACM Trans.\ Gr.}
\renewcommand*{\DTLtoms}{ACM Trans.\ Math. Softw.}
\renewcommand*{\DTLtoois}{ACM Trans. Office Inf.\ Syst.}
\renewcommand*{\DTLtoplas}{ACM Trans.\ Prog. Lang.\ Syst.}
\renewcommand*{\DTLtcs}{Theoretical Comput.\ Sci.}
```

End of 'abbrv' style.

```
}
```

\dtlbst@alpha Define 'alpha' style. This is similar to 'plain' except that the labels are strings rather than numerical.

```
\newcommand{\dtlbst@alpha}{%
```

Base this style on 'plain':

```
\dtlbst@plain
```

Set how to format the entire bibliography:

```
\renewenvironment{DTLthebibliography}[2][\boolean{true}]{%
\dtl@createalphabiblabels{##1}{##2}%
\begin{thebibliography}{\@dtl@widestlabel}%
}{\end{thebibliography}}%
```

Set how to start the bibliography entry:

```
\renewcommand*{\DTLbibitem}{%
\expandafter\bibitem\expandafter
 [\csname dtl@biblabel@\DBIBcitekey\endcsname]{\DBIBcitekey}}%
\renewcommand*{\DTLmbibitem}[1]{%
\expandafter\bibitem\expandafter
 [\csname dtl@biblabel@\DBIBcitekey\endcsname]{##1@\DBIBcitekey}}%
```

End of ‘alpha’ style.

```
}
```

tealphabiblabels

```
\dtl@createalphabiblabels{⟨condition⟩}{⟨db name⟩}
```

Constructs the alpha style bib labels for the given database. (Labels are stored in the control sequence `\dtl@biblabel@`⟨*citekey*⟩.) This also sets `\@dtl@widestlabel` to the widest label.

```
\newcommand*{\dtl@createalphabiblabels}[2]{%
\dtl@message{Creating bib labels}%
\begingroup
\gdef\@dtl@widestlabel{}%
\dtl@widest=0pt\relax
\DTLforeachbibentry[#1]{#2}{%
\dtl@message{\DBIBcitekey}%
\DTLifbibfieldexists{Author}{%
  \dtl@listgetalphalabel{\@dtl@thislabel}{\@dtl@key@Author}%
}{%
\DTLifbibfieldexists{Editor}{%
    \dtl@listgetalphalabel{\@dtl@thislabel}{\@dtl@key@Editor}%
 }{%
   \DTLifbibfieldexists{Key}{%
     \expandafter\dtl@get@firstthree\expandafter
       {\@dtl@key@Key}{\@dtl@thislabel}%
   }{%
     \DTLifbibfieldexists{Organization}{%
       \expandafter\dtl@get@firstthree\expandafter
         {\@dtl@key@Organization}{\@dtl@thislabel}%
   }{%
       \expandafter\dtl@get@firstthree\expandafter
         {\DBIBentrytype}{\@dtl@thislabel}%
   }%
   }}}%
\DTLifbibfieldexists{Year}{}{\DTLifbibfieldexists{CrossRef}{%
\DTLgetvalueforkey{\@dtl@key@Year}{Year}{#2}{CiteKey}{%
\@dtl@key@CrossRef}}{}}%
\DTLifbibfieldexists{Year}{%
\expandafter\dtl@get@yearsuffix\expandafter{\@dtl@key@Year}%
\expandafter\toks@\expandafter{\@dtl@thislabel}%
\expandafter\@dtl@toks\expandafter{\@dtl@year}%
\edef\@dtl@thislabel{\the\toks@\the\@dtl@toks}%
}{}%
\let\@dtl@s@thislabel=\@dtl@thislabel
\@onelevel@sanitize\@dtl@s@thislabel
\@ifundefined{c@biblabel@\@dtl@s@thislabel}{%
\newcounter{biblabel@\@dtl@s@thislabel}%
\setcounter{biblabel@\@dtl@s@thislabel}{1}%
\expandafter\edef\csname @dtl@bibfirst@\@dtl@s@thislabel\endcsname{%
\DBIBcitekey}%
\expandafter\global
\expandafter\let\csname dtl@biblabel@\DBIBcitekey\endcsname=
  \@dtl@thislabel
}{%
\expandafter\ifnum\csname c@biblabel@\@dtl@s@thislabel\endcsname=1\relax
 \expandafter\let\expandafter\@dtl@tmp
   \csname @dtl@bibfirst@\@dtl@s@thislabel\endcsname
 \expandafter\protected@xdef\csname dtl@biblabel@\@dtl@tmp\endcsname{%
   \@dtl@thislabel a}%
\fi
\stepcounter{biblabel@\@dtl@s@thislabel}%
\expandafter\protected@xdef\csname dtl@biblabel@\DBIBcitekey\endcsname{%
   \@dtl@thislabel\alph{biblabel@\@dtl@s@thislabel}}%
}%
\settowidth{\dtl@tmplength}{%
 \csname dtl@biblabel@\DBIBcitekey\endcsname}%
\ifdim\dtl@tmplength>\dtl@widest
 \dtl@widest=\dtl@tmplength
 \expandafter\global\expandafter\let\expandafter\@dtl@widestlabel
  \expandafter=\csname dtl@biblabel@\DBIBcitekey\endcsname
\fi
}%
\endgroup
}
```

stgetalphalabel Determine the alpha style label from a list of authors/editors (the first argument must be a control sequence (in which the label is stored), the second argument must be the list of names.)

```
\newcommand*{\dtl@listgetalphalabel}[2]{%
\@dtl@authorcount=0\relax
\@for\@dtl@author:=#2\do{%
\advance\@dtl@authorcount by 1\relax}%
\ifnum\@dtl@authorcount=1\relax
 \expandafter\dtl@getsinglealphalabel#2{#1}\relax
\else
  {%
   \xdef#1{}%
  \@dtl@tmpcount=0\relax
   \def\DTLafterinitials{}\def\DTLbetweeninitials{}%
   \def\DTLafterinitialbeforehyphen{}\def\DTLinitialhyphen{}%
   \@for\@dtl@author:=#2\do{%
     \expandafter\dtl@getauthorinitial\@dtl@author
     \expandafter\toks@\expandafter{\@dtl@tmp}%
     \expandafter\@dtl@toks\expandafter{#1}%
     \xdef#1{\the\@dtl@toks\the\toks@}%
```

```
      \advance\@dtl@tmpcount by 1\relax
      \ifnum\@dtl@tmpcount>2\relax\@endfortrue\fi
   }}%
\fi
}
```

Get author's initial (stores in `\@dtl@tmp`):

```
\newcommand*{\dtl@getauthorinitial}[4]{%
\def\@dtl@vonpart{#1}%
\ifx\@dtl@vonpart\@empty
 \DTLstoreinitials{#2}{\@dtl@tmp}%
\else
 \DTLstoreinitials{#1 #2}{\@dtl@tmp}%
\fi}
```

Get label for single author (last argument is control sequence in which to store the label):

```
\newcommand*{\dtl@getsinglealphalabel}[5]{%
\def\@dtl@vonpart{#1}%
\ifx\@dtl@vonpart\@empty
 \DTLifSubString{#2}{-}{%
   {\def\DTLafterinitials{}\def\DTLbetweeninitials{}%
    \def\DTLafterinitialbeforehyphen{}%
    \def\DTLinitialhyphen{}%
    \DTLstoreinitials{#2}{\@dtl@tmp}\global\let#5=\@dtl@tmp}%
   }{%
   \dtl@getfirstthree{#5}#2{}{}{}{}\@nil
 }
\else
 {\def\DTLafterinitials{}\def\DTLbetweeninitials{}%
  \def\DTLafterinitialbeforehyphen{}%
  \def\DTLinitialhyphen{}%
  \DTLstoreinitials{#1 #2}{\@dtl@tmp}\global\let#5=\@dtl@tmp}%
\fi
}
```

Get first three letters from the given string:

```
\def\dtl@getfirstthree#1#2#3#4#5\@nil{%
  \def#1{#2#3#4}%
}
\newcommand*{\dtl@get@firstthree}[2]{%
\dtl@getfirstthree#2#1{}{}{}{}{}\@nil}
```

Get year suffix:

```
\newcommand*{\dtl@get@yearsuffix}[1]{%
\dtl@getyearsuffix#1\@nil\relax\relax}

\def\dtl@getyearsuffix#1#2#3{%
\def\@dtl@argi{#1}\def\@dtl@argii{#2}%
\def\@dtl@argiii{#3}%
\ifx\@dtl@argi\@nnil
```

```
 \def\@dtl@year{}%
 \let\@dtl@donext=\relax
\else
 \ifx\@dtl@argii\@nnil
  \dtl@ifsingle{#1}{%
   \def\@dtl@year{#1}%
   \let\@dtl@donext=\relax
  }{%
   \def\@dtl@donext{\dtl@getyearsuffix#1#2#3}%
  }%
 \else
  \ifx\@dtl@argiii\@nnil
   \dtl@ifsingle{#1}{%
    \dtl@ifsingle{#2}{%
     \def\@dtl@year{#1#2}%
     \let\@dtl@donext=\relax
    }{%
     \def\@dtl@donext{\dtl@getyearsuffix#2#3}%
    }%
   }{%
    \def\@dtl@donext{\dtl@getyearsuffix#2#3}%
   }%
  \else
   \def\@dtl@donext{\dtl@getyearsuffix{#2}{#3}}%
  \fi
 \fi
\fi
\@dtl@donext
}
```

ibliographystyle

\DTLbibliographystyle{⟨*style*⟩}

Sets the bibliography style.

```
\newcommand*{\DTLbibliographystyle}[1]{%
\@ifundefined{dtlbst@#1}{\PackageError{databib}{Unknown
bibliography style ‘#1’}{}}{\csname dtlbst@#1\endcsname}}
```

Set the default bibliography style:

```
\DTLbibliographystyle{\dtlbib@style}
```

6.8 Multiple Bibliographies

In order to have multiple bibliographies, there needs to be an aux file for each bibliography. The main bibliography is in `\jobname`.aux, but need to provide a means of creating additional aux files.

\DTLmultibibs

\DTLmultibibs{⟨*list*⟩}

This creates an auxiliary file for each name in ⟨*list*⟩. For example, `\DTLmultibibs{foo,bar}` will create the files `foo.aux` and `bar.aux`.

```
\newcommand*{\DTLmultibibs}[1]{%
\@for\@dtl@af:=#1\do{%
\@ifundefined{dtl@aux@\@dtl@af}{%
\expandafter\newwrite\csname dtl@aux@\@dtl@af\endcsname
\expandafter\immediate
\expandafter\openout\csname dtl@aux@\@dtl@af\endcsname=\@dtl@af.aux
\expandafter\def\csname b@\@dtl@af @*\endcsname{}%
}{%
\PackageError{databib}{Can't create auxiliary file '\@dtl@af.aux',
\expandafter\string\csname dtl@aux@\@dtl@af\endcsname\space
already exists}{}}}}
```

Can only be used in the preamble:

```
\@onlypreamble{\DTLmultibibs}
```

\DTLcite

\DTLcite[⟨*text*⟩]{⟨*mbib*⟩}{⟨*labels*⟩}

This is similar to `\cite[`⟨*text*⟩`]{`⟨*labels*⟩`}`, except 1) the cite information is written to the auxiliary file associated with the multi-bib ⟨*mbib*⟩ (which must be named in `\DTLmultibibs`) and 2) the cross referencing label is constructed from ⟨*mbib*⟩ and ⟨*label*⟩ to allow for the same citation to appear in multiple bibliographies.

```
\newcommand*{\DTLcite}{\@ifnextchar[{\@tempswatrue \dtl@citex
}{\@tempswafalse \dtl@citex[]}}
```

\dtl@citex

```
\def\dtl@citex[#1]#2#3{%
\leavevmode\let\@citea\@empty
\@cite{\@for\@citeb:=#3\do{\@citea
  \def\@citea{,\penalty \@m \ }%
  \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
  \if@filesw
    \@ifundefined{dtl@aux@#2}{%
      \PackageError{databib}{multibib '#2' not defined}{%
      You need to define '#2' in \string\DTLmutlibibs}%
    }{%
      \expandafter\immediate
      \expandafter\write\csname dtl@aux@#2\endcsname{%
        \string\citation{\@citeb}}%
    }%
  \fi
  \@ifundefined{b@#2@\@citeb}{%
    \hbox{\reset@font\bfseries ?}%
    \G@refundefinedtrue
    \@latex@warning{Citation ‘\@citeb ’ on page \thepage \space
      undefined}%
  }{%
    \@cite@ofmt{\csname b@#2@\@citeb \endcsname }%
  }%
}}{#1}%
}
```

\DTLnocite

\DTLnocite{⟨*mbib*⟩}{⟨*key list*⟩}

As \nocite but uses the aux file associated with ⟨*mbib*⟩ which must have been defined using \DTLmultibibs.

```
\newcommand*{\DTLnocite}[2]{%
\@ifundefined{dtl@aux@#1}{%
  \PackageError{databib}{multibib ‘#1’ not defined}{%
  You need to define ‘#1’ in \string\DTLmutlibibs}%
}{%
  \@bsphack
  \ifx\@onlypreamble\document
    \@for\@citeb:=#2\do{%
      \edef\@citeb{\expandafter\@firstofone\@citeb}%
      \if@filesw
        \expandafter\immediate
        \expandafter\write\csname dtl@aux@#1\endcsname{%
          \string\citation{\@citeb}}%
      \fi
      \@ifundefined{b@#1@\@citeb}{%
        \G@refundefinedtrue
        \@latex@warning{Citation ‘\@citeb ’ undefined}}{}%
    }%
  \else
    \@latex@error{Cannot be used in preamble}\@eha
  \fi
  \@esphack
}%
}
```

\DTLloadmbbl

\DTLloadmbib{⟨*mbib*⟩}{⟨*db name*⟩}{⟨*bib list*⟩}

```
\newcommand*{\DTLloadmbbl}[3]{%
\@ifundefined{dtl@aux@#1}{%
```

```
  \PackageError{databib}{multibib ‘#1’ not defined}{%
  You need to define ‘#1’ in \string\DTLmutlibibs}%
}{%
  \if@filesw
    \expandafter\immediate\expandafter
      \write\csname dtl@aux@#1\endcsname{\string\bibstyle{databib}}%
    \expandafter\immediate\expandafter
      \write\csname dtl@aux@#1\endcsname{\string\bibdata{#3}}%
  \fi
  \DTLnewdb{#2}%
  \edef\DTLBIBdbname{#2}%
  \@input@{#1.bbl}%
}%
}
```

\DTLmbibliography[⟨*condition*⟩]{⟨*mbib name*⟩}{⟨*bib dbname*⟩}

Displays the bibliography for the database ⟨*bib dbname*⟩ which must have previously been loaded using \DTLloadmbbl, where ⟨*mbib name*⟩ must be listed in \DTLmultibibs.

TLmbibliography

```
\newcommand*{\DTLmbibliography}[3][\boolean{true}]{%
\begin{DTLthebibliography}[#1]{#3}%
\DTLforeachbibentry[#1]{#3}{%
\DTLmbibitem{#2} \DTLformatbibentry \DTLendbibitem
}%
\end{DTLthebibliography}%
}
```

7 databar.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{databar}[2019/09/27 v2.32 (NLCT)]
```

Require xkeyval package

```
\RequirePackage{xkeyval}
```

Require dataplot package

```
\RequirePackage{dataplot}
```

TLcolorbarchart The conditional \ifDTLcolorbarchart is used to determine whether to use colour or grey scale.

```
\newif\ifDTLcolorbarchart
\DTLcolorbarcharttrue
```

Package options to change the conditional:

```
\DeclareOption{color}{\DTLcolorbarcharttrue}
\DeclareOption{gray}{\DTLcolorbarchartfalse}
```

\DTLbarXlabelalign specifies the alignment for the *x* axis labels.

```
\newcommand*{\DTLbarXlabelalign}{left,rotate=-90}
```

\DTLbarYticklabelalign specifies the alignment for the *x* axis labels.

```
\newcommand*{\DTLbarYticklabelalign}{right}
```

DTLverticalbars Define boolean keys to govern bar chart orientation.

```
\define@boolkey{databar}[DTL]{verticalbars}[true]{%
\ifDTLverticalbars
 \def\DTLbarXlabelalign{left,rotate=-90}%
 \def\DTLbarYticklabelalign{right}
\else
 \def\DTLbarXlabelalign{right}%
 \def\DTLbarYticklabelalign{center}
\fi}
```

Set defaults:

```
\DTLverticalbarstrue
```

Package options to change \ifDTLverticalbars

```
\DeclareOption{vertical}{\DTLverticalbarstrue
 \def\DTLbarXlabelalign{left,rotate=-90}%
 \def\DTLbarYticklabelalign{right}
}
\DeclareOption{horizontal}{\DTLverticalbarsfalse
 \def\DTLbarXlabelalign{right}%
 \def\DTLbarYticklabelalign{center}
}
```

Process options:

```
\ProcessOptions
```

Required packages:

```
\RequirePackage{datatool}
\RequirePackage{tikz}
```

Define some variables that govern the appearance of the bar chart.

Lbarchartlength The total height of the bar chart is given by \DTLbarchartheight

```
\newlength\DTLbarchartlength
\DTLbarchartlength=3in
```

\DTLbarwidth The width of each bar is given by \DTLbarwidth.

```
\newlength\DTLbarwidth
\DTLbarwidth=1cm
```

Lbarlabeloffset The offset from the *x* axis to the bar label if given by \DTLbarlabeloffset.

```
\newlength\DTLbarlabeloffset
\setlength\DTLbarlabeloffset{10pt}
```

TLBarXAxisStyle The style of the *x* axis is given by \DTLBarXAxisStyle

```
\newcommand*{\DTLBarXAxisStyle}{-}
```

TLBarYAxisStyle The style of the *y* axis is given by \DTLBarYAxisStyle.

```
\newcommand*{\DTLBarYAxisStyle}{-}
```

DTLbarroundvar DTLbarroundvar is a counter governing the number of digits to round to when using \FPround.

```
\newcounter{DTLbarroundvar}
\setcounter{DTLbarroundvar}{1}
```

splayYticklabel \DTLbardisplayYticklabel governs how the *y* tick labels appear.

```
\newcommand*{\DTLbardisplayYticklabel}[1]{#1}
```

aylowerbarlabel \DTLdisplaylowerbarlabel governs how the lower bar labels appear.

```
\newcommand*{\DTLdisplaylowerbarlabel}[1]{#1}
```

ermultibarlabel \DTLdisplaylowermultibarlabel governs how the lower multi bar labels appear.

```
\newcommand*{\DTLdisplaylowermultibarlabel}[1]{#1}
```

ayupperbarlabel \DTLdisplayupperbarlabel governs how the upper bar labels appear.

```
\newcommand*{\DTLdisplayupperbarlabel}[1]{#1}
```

ermultibarlabel \DTLdisplayuppermultibarlabel governs how the upper multi bar labels appear.

```
\newcommand*{\DTLdisplayuppermultibarlabel}[1]{#1}
```

Lbaratbegintikz \DTLbaratbegintikz specifies any commands to apply at the start of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLbaratbegintikz}{}
```

DTLbaratendtikz \DTLbaratendtikz specifies any commands to apply at the end of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLbaratendtikz}{}
```

\ifDTLbarxaxis The conditional \ifDTLbarxaxis is used to determine whether or not to display the *x* axis

```
\newif\ifDTLbarxaxis
```

\ifDTLbaryaxis The conditional \ifDTLbaryaxis is used to determine whether or not to display the *y* axis.

```
\newif\ifDTLbaryaxis
```

\ifDTLbarytics The conditional \ifDTLbarytics to determine whether or not to display the *y* tick marks.

```
\newif\ifDTLbarytics
```

\@dtl@barcount The count register \@dtl@barcount is used to store the current bar index.

```
\newcount\@dtl@barcount
```

\DTLsetbarcolor

\DTLsetbarcolor{⟨*n*⟩}{⟨*color*⟩}

Assigns colour name ⟨*color*⟩ to the ⟨*n*⟩th bar.

```
\newcommand*{\DTLsetbarcolor}[2]{%
\expandafter\def\csname dtlbar@segcol\romannumeral#1\endcsname{#2}%
}
```

\DTLgetbarcolor

\DTLgetbarcolor{⟨*n*⟩}

Gets the colour specification for the ⟨*n*⟩th bar.

```
\newcommand*{\DTLgetbarcolor}[1]{%
\csname dtlbar@segcol\romannumeral#1\endcsname}
```

\DTLdobarcolor

\DTLdobarcolor{⟨*n*⟩}

Sets the colour to that for the ⟨*n*⟩th bar.

```
\newcommand*{\DTLdobarcolor}[1]{%
\expandafter\color\expandafter
{\csname dtlbar@segcol\romannumeral#1\endcsname}}
```

currentbarcolor \DTLdocurrentbarcolor sets the colour to that of the current bar.

```
\newcommand*{\DTLdocurrentbarcolor}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{databar}{Can't use
  \string\DTLdocurrentbarcolor\space outside
  \string\DTLbarchart}{}%
\else
  \expandafter\DTLdobarcolor\expandafter{%
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
\fi}
```

baroutlinecolor \DTLbaroutlinecolor specifies what colour to draw the outline.

```
\newcommand*{\DTLbaroutlinecolor}{black}
```

baroutlinewidth \DTLbaroutlinewidth specifies the line width of the outline: Outline is only drawn if the linewidth is greater than 0pt.

```
\newlength\DTLbaroutlinewidth
\DTLbaroutlinewidth=0pt
```

Set the default colours. If there are more than eight bars, more colours will need to be defined.

```
\ifDTLcolorbarchart
 \DTLsetbarcolor{1}{red}
 \DTLsetbarcolor{2}{green}
 \DTLsetbarcolor{3}{blue}
 \DTLsetbarcolor{4}{yellow}
 \DTLsetbarcolor{5}{magenta}
 \DTLsetbarcolor{6}{cyan}
 \DTLsetbarcolor{7}{orange}
 \DTLsetbarcolor{8}{white}
\else
 \DTLsetbarcolor{1}{black!15}
 \DTLsetbarcolor{2}{black!25}
 \DTLsetbarcolor{3}{black!35}
 \DTLsetbarcolor{4}{black!45}
 \DTLsetbarcolor{5}{black!55}
 \DTLsetbarcolor{6}{black!65}
 \DTLsetbarcolor{7}{black!75}
 \DTLsetbarcolor{8}{black!85}
\fi
```

DTLeverybarhook Code to apply at every bar. The start point of the bar can be accessed via \DTLstartpt, the mid point of the bar can be accessed via \DTLmidpt and the end point of the bar can be accessed via \DTLendpt

```
\newcommand*{\DTLeverybarhook}{}
```

Define keys for \DTLbarchart optional argument. Set the maximum value of the y axis.

```
\define@key{databar}{max}{\def\DTLbarmax{#1}}
```

Set the total length of the bar chart

```
\define@key{databar}{length}{\DTLbarchartlength=#1\relax
}
```

Set the maximum depth (negative extent)

```
\define@key{databar}{maxdepth}{%
\ifnum#1>0\relax
 \PackageError{databar}{depth must be zero or negative}{}%
\else
 \def\DTLnegextent{#1}%
\fi}
```

Determine which axes should be shown

```
\define@choicekey{databar}{axes}[\var\nr]{both,x,y,none}{%
\ifcase\nr\relax
 % both
  \DTLbarxaxistrue
  \DTLbaryaxistrue
  \DTLbaryticstrue
\or
 % x only
  \DTLbarxaxistrue
  \DTLbaryaxisfalse
  \DTLbaryticsfalse
\or
 % y only
  \DTLbarxaxisfalse
  \DTLbaryaxistrue
  \DTLbaryticstrue
\or
 % neither
  \DTLbarxaxisfalse
  \DTLbaryaxisfalse
  \DTLbaryticsfalse
\fi
}
```

Variable used to create the bar chart. (Must be a control sequence.)

```
\define@key{databar}{variable}{%
 \def\DTLbarvariable{#1}%
}
```

Variables used to create the multi bar chart. (Must be a comma separated list of control sequences.)

```
\define@key{databar}{variables}{%
 \def\dtlbar@variables{#1}%
}
```

Bar width

```
\define@key{databar}{barwidth}{\setlength{\DTLbarwidth}{#1}}
```

Lower bar labels

```
\define@key{databar}{barlabel}{%
\def\dtl@barlabel{#1}}
\def\dtl@barlabel{}
```

Lower bar labels for multi-bar charts

```
\define@key{databar}{multibarlabels}{%
\def\dtl@multibarlabels{#1}}
\def\dtl@multibarlabels{}
```

Gap between groups in multi-bar charts (This should be in x units where 1 x unit is the width of a bar.)

```
\define@key{databar}{groupgap}{\def\dtlbar@groupgap{#1}}
\def\dtlbar@groupgap{1}
```

Upper bar labels

```
\define@key{databar}{upperbarlabel}{%
\def\dtl@upperbarlabel{#1}}
\def\dtl@upperbarlabel{}
```

Upper bar labels for multi-bar charts

```
\define@key{databar}{uppermultibarlabels}{%
\def\dtl@uppermultibarlabels{#1}}
\def\dtl@uppermultibarlabels{}
```

Define list of points for y tics. (Must be a comma separated list of decimal numbers.)

```
\define@key{databar}{yticpoints}{%
\def\dtlbar@yticlist{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticlist=\relax
```

Set the y tick gap:

```
\define@key{databar}{yticgap}{%
\def\dtlbar@yticgap{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticgap=\relax
```

Define list of labels for y tics.

```
\define@key{databar}{yticlabels}{%
\def\dtlbar@yticlabels{#1}\DTLbaryticstrue\DTLbaryaxistrue}
\let\dtlbar@yticlabels=\relax
```

Define y axis label.

```
\define@key{databar}{ylabel}{%
\def\dtlbar@ylabel{#1}}
\let\dtlbar@ylabel=\relax
```

\DTLbarchart

```
\DTLbarchart[⟨conditions⟩]{⟨option list⟩}{⟨db name⟩}{⟨assign list⟩}
```

Make a bar chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include `variable=`⟨*cmd*⟩, where ⟨*cmd*⟩ is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

```
\newcommand*{\DTLbarchart}[4][\boolean{true}]{%
{%
  \undef\DTLbarvariable
  \undef\DTLbarmax
  \undef\DTLnegextent
  \disable@keys{databar}{variables,multibarlabels,%
    uppermultibarlabels,groupgap}%
  \setkeys{databar}{#2}%
  \ifundef{\DTLbarvariable}%
  {%
    \PackageError{databar}%
    {\string\DTLbarchart\space missing variable}%
    {You haven't use the "variable" key}%
  }%
  {%
```

Compute the maximum bar height, unless `\DTLbarmax` has been set.

```
    \ifundef{\DTLbarmax}%
    {%
      \@sDTLforeach[#1]{#3}{#4}{%
        \expandafter\DTLconverttodecimal\expandafter
          {\DTLbarvariable}{\dtl@barvar}%
        \ifundef{\DTLbarmax}%
        {%
          \let\DTLbarmax=\dtl@barvar
        }%
        {%
          \let\dtl@old=\DTLbarmax
          \dtlmax{\DTLbarmax}{\dtl@old}{\dtl@barvar}%
        }%
      }%
      \ifx\dtlbar@yticgap\relax
      \else
        \let\dtl@thistick=\dtlbar@yticgap
        \whiledo{\DTLisFPopenbetween{\dtl@thistick}{0}{\DTLbarmax}}%
        {%
          \dtladd{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
        }%
        \let\DTLbarmax=\dtl@thistick
      \fi
    }%
    {}%
```

Compute the bar depth, unless `\DTLnegextent` has been set.

```
    \ifundef{\DTLnegextent}%
    {%
```

```
  \def\DTLnegextent{0}%
  \@sDTLforeach[#1]{#3}{#4}{%
    \expandafter\DTLconverttodecimal\expandafter
      {\DTLbarvariable}{\dtl@barvar}%
    \let\dtl@old=\DTLnegextent
    \DTLmin{\DTLnegextent}{\dtl@old}{\dtl@barvar}%
  }%
  \ifx\dtlbar@yticgap\relax
  \else
    \ifthenelse{\DTLisFPlt{\DTLnegextent}{0}}%
    {%
    \edef\dtl@thistick{0}%
    \whiledo{\DTLisFPclosedbetween{\dtl@thistick}{\DTLnegextent}{0}}{%
      \dtlsub{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
    }%
    \let\DTLnegextent=\dtl@thistick
    }{}%
  \fi
}%
{}%
```

Determine scaling factor

```
\@dtl@tmpcount=\DTLbarchartlength
\dtlsub{\dtl@extent}{\DTLbarmax}{\DTLnegextent}%
\dtldiv{\dtl@unit}{\number\@dtl@tmpcount}{\dtl@extent}%
```

Construct y tick list if required

```
 \setlength{\dtl@yticlabelwidth}{0pt}%
 \ifDTLbarytics
   \ifx\dtlbar@yticlist\relax
     \ifx\dtlbar@yticgap\relax
       \@dtl@tmpcount=\DTLmintickgap
       \divide\@dtl@tmpcount by 65536\relax
       \dtldiv{\dtl@mingap}{\number\@dtl@tmpcount}{\dtl@unit}%
       \dtl@constructticklist\DTLnegextent\DTLbarmax
         \dtl@mingap\dtlbar@yticlist
     \else
       \dtl@constructticklistwithgapz
         \DTLnegextent\DTLbarmax\dtlbar@yticlist\dtlbar@yticgap
     \fi
   \fi
   \ifx\dtlbar@ylabel\relax
   \else
     \ifx\dtlbar@yticlabels\relax
       \@for\dtl@thislabel:=\dtlbar@yticlist\do{%
         \dtlround{\dtl@thislabel}{\dtl@thislabel}
                 {\c@DTLbarroundvar}%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
         \else
           \settoheight{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \edef\@dtl@h{\the\dtl@tmplength}%
           \settodepth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \addtolength{\dtl@tmplength}{\@dtl@h}%
           \addtolength{\dtl@tmplength}{\baselineskip}%
         \fi
         \ifdim\dtl@tmplength>\dtl@yticlabelwidth
           \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
         \fi
       }%
     \else
       \@for\dtl@thislabel:=\dtlbar@yticlabels\do{%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
             \DTLbardisplayYticklabel{\dtl@thislabel}}%
         \else
           \settoheight{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \edef\@dtl@h{\the\dtl@tmplength}%
           \settodepth{\dtl@tmplength}{%
              \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \addtolength{\dtl@tmplength}{\@dtl@h}%
           \addtolength{\dtl@tmplength}{\baselineskip}%
         \fi
         \ifdim\dtl@tmplength>\dtl@yticlabelwidth
           \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
         \fi
       }%
     \fi
   \fi
 \fi
```

Store the width of the bar chart in \DTLbarchartwidth

```
\edef\DTLbarchartwidth{\expandafter\number\csname dtlrows@#3\endcsname}
```

Do the bar chart

```
\begin{tikzpicture}
```

Set unit vectors

```
\ifDTLverticalbars
  \pgfsetyvec{\pgfpoint{0pt}{\dtl@unit sp}}%
  \pgfsetxvec{\pgfpoint{\DTLbarwidth}{0pt}}%
\else
  \pgfsetxvec{\pgfpoint{\dtl@unit sp}{0pt}}%
  \pgfsetyvec{\pgfpoint{0pt}{\DTLbarwidth}}%
\fi
```

Begin hook

```
\DTLbaratbegintikz
```

Initialise

```
\def\@dtl@start{0}%
```

Iterate through data

```
\@sDTLforeach[#1]{#3}{#4}{%
```

Store the bar number in \@dtl@bar

```
\expandafter\let\expandafter\@dtl@bar
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
```

Convert variable to decimal

```
\expandafter\DTLconverttodecimal\expandafter
  {\DTLbarvariable}{\dtl@variable}%
```

Draw bars

```
\begin{scope}
 \DTLdocurrentbarcolor
 \ifDTLverticalbars
   \fill (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@bar,\dtl@variable) -- (\@dtl@bar,0) -- cycle;
 \else
   \fill (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@bar) -- (0,\@dtl@bar) -- cycle;
 \fi
\end{scope}
```

Draw outline

```
\begin{scope}
\ifdim\DTLbaroutlinewidth>0pt
 \expandafter\color\expandafter{\DTLbaroutlinecolor}
 \ifDTLverticalbars
   \draw (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@bar,\dtl@variable) -- (\@dtl@bar,0) -- cycle;
 \else
   \draw (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@bar) -- (0,\@dtl@bar) -- cycle;
 \fi
\fi
\end{scope}
```

Draw lower *x* labels

```
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@start.5}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}},
     \DTLbarXlabelalign
  }%
```

Set \DTLstartpt to the starting point.

```
      \edef\DTLstartpt{\noexpand\pgfpointxy{\@dtl@start.5}{0}}%
    \else
      \edef\dtl@textopt{%
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{0}{\@dtl@start.5}}
               {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}},
         \DTLbarXlabelalign
      }%
```

Set \DTLstartpt to the starting point.

```
      \edef\DTLstartpt{\noexpand\pgfpointxy{0}{\@dtl@start.5}}%
    \fi
     \expandafter\pgftext\expandafter[\dtl@textopt]{%
       \DTLdisplaylowerbarlabel{\dtl@barlabel}}
```

Draw upper *x* labels

```
    \ifDTLverticalbars
```

Vertical bars

```
      \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}%
      {
        \edef\dtl@textopt{%
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}
               {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}}
        }%
      }{%
        \edef\dtl@textopt{%
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}
               {\noexpand\pgfpoint{0pt}{\noexpand\DTLbarlabeloffset}}}
        }%
      }
```

Set \DTLendpt to the end point.

```
      \edef\DTLendpt{\noexpand\pgfpointxy{\@dtl@start.5}{\dtl@variable}}%
    \else
```

Horizontal bars

```
      \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}%
      {
        \edef\dtl@textopt{right,
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}
               {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}}
        }%
      }{%
        \edef\dtl@textopt{left,
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}
               {\noexpand\pgfpoint{\noexpand\DTLbarlabeloffset}{0pt}}}
```

```
    }%
  }
```

Set \DTLendpt to the end point.

```
  \edef\DTLendpt{\noexpand\pgfpointxy{\dtl@variable}{\@dtl@start.5}}%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
    \DTLdisplayupperbarlabel{\dtl@upperbarlabel}}
```

Set the mid point

```
\def\DTLmidpt{\pgfpointlineattime{0.5}{\DTLstartpt}{\DTLendpt}}%
```

Do every bar hook

```
  \DTLeverybarhook
```

End of loop

```
  \edef\@dtl@start{\number\@dtl@bar}%
}%
```

Draw *x* axis

```
\ifDTLbarxaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (\DTLbarchartwidth,0);
  \else
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (0,\DTLbarchartwidth);
  \fi
\fi
```

Draw *y* axis

```
\ifDTLbaryaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (0,\DTLnegextent) -- (0,\DTLbarmax);
  \else
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (\DTLnegextent,0) -- (\DTLbarmax,0);
  \fi
\fi
```

Plot *y* tick marks if required

```
\ifx\dtlbar@yticlist\relax
\else
  \@for\dtl@thistick:=\dtlbar@yticlist\do{%
    \ifDTLverticalbars
      \pgfpathmoveto{\pgfpointxy{0}{\dtl@thistick}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{0}{\dtl@thistick}}
                    {\pgfpoint{-\DTLticklength}{0pt}}}
    \else
      \pgfpathmoveto{\pgfpointxy{\dtl@thistick}{0}}
      \pgfpathlineto{
        \pgfpointadd{\pgfpointxy{\dtl@thistick}{0}}
                    {\pgfpoint{0pt}{-\DTLticklength}}}
    \fi
    \pgfusepath{stroke}
    \ifx\dtlbar@yticlabels\relax
       \dtlround{\dtl@thislabel}{\dtl@thistick}
               {\c@DTLbarroundvar}%
    \else
       \dtl@chopfirst\dtlbar@yticlabels\dtl@thislabel\dtl@rest
       \let\dtlbar@yticlabels=\dtl@rest
    \fi
    \ifDTLverticalbars
      \edef\dtl@textopt{\DTLbarYticklabelalign,%
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{0}{\dtl@thistick}}
             {\noexpand\pgfpoint{-\noexpand\DTLticklabeloffset}{0pt}},
        }}%
    \else
      \edef\dtl@textopt{\DTLbarYticklabelalign,
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\dtl@thistick}{0}}
             {\noexpand\pgfpoint{0pt}{-\noexpand\DTLticklabeloffset}}
        }}%
    \fi
    \expandafter\pgftext\expandafter[\dtl@textopt]{%
      \DTLbardisplayYticklabel{\dtl@thislabel}}
  }%
\fi
```

Plot the *y* label if required

```
\ifx\dtlbar@ylabel\relax
\else
  \addtolength{\dtl@yticlabelwidth}{\baselineskip}%
  \setlength{\dtl@tmplength}{0.5\DTLbarchartlength}
  \ifDTLverticalbars
    \pgftext[bottom,center,at={\pgfpointadd
        {\pgfpointxy{0}{\DTLnegextent}}%
        {\pgfpoint{-\dtl@yticlabelwidth}{\dtl@tmplength}}},
        rotate=90]{%
       \dtlbar@ylabel}
  \else
    \pgftext[bottom,center,at={\pgfpointadd
        {\pgfpointxy{\DTLnegextent}{0}}%
        {\pgfpoint{\dtl@tmplength}{-\dtl@yticlabelwidth}}}]{%
       \dtlbar@ylabel}
  \fi
\fi
```

Finish bar chart

```
    \DTLbaratendtikz
    \end{tikzpicture}
    }%
  }%
  }
```

DTLmultibarchart

`\DTLmultibarchart[⟨conditions⟩]{⟨option list⟩}{⟨db name⟩}{⟨assign list⟩}`

Make a multi-bar chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include the `variables` key which must be a comma separated list of commands, where each command is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

```
  \newcommand*{\DTLmultibarchart}[4][\boolean{true}]{%
  {\let\dtlbar@variables=\relax
  \let\DTLbarmax=\relax
  \let\DTLnegextent=\relax
  \disable@keys{databar}{variable,upperbarlabel}%
  \setkeys{databar}{#2}%
  \ifx\dtlbar@variables\relax
    \PackageError{databar}{\string\DTLmultibarchart\space missing variables setting}{}%
  \else
```

Compute the maximum bar height, unless `\DTLbarmax` has been set.

```
    \ifx\DTLbarmax\relax
      \@sDTLforeach[#1]{#3}{#4}{%
        \@for\DTLbarvariable:=\dtlbar@variables\do{%
          \expandafter\DTLconverttodecimal\expandafter
            {\DTLbarvariable}{\dtl@barvar}%
          \ifx\DTLbarmax\relax
            \let\DTLbarmax=\dtl@barvar
          \else
            \let\dtl@old=\DTLbarmax
            \dtlmax{\DTLbarmax}{\dtl@old}{\dtl@barvar}%
          \fi
        }%
      }%
      \ifx\dtlbar@yticgap\relax
      \else
        \let\dtl@thistick=\dtlbar@yticgap%
        \whiledo{\DTLisFPopenbetween{\dtl@thistick}{0}{\DTLbarmax}}{%
          \dtladd{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
        }%
        \let\DTLbarmax=\dtl@thistick
      \fi
    \fi
```

Compute the bar depth, unless `\DTLnegextent` has been set.

```
\ifx\DTLnegextent\relax
  \def\DTLnegextent{0}%
  \@sDTLforeach[#1]{#3}{#4}{%
    \@for\DTLbarvariable:=\dtlbar@variables\do{%
      \expandafter\DTLconverttodecimal\expandafter
        {\DTLbarvariable}{\dtl@barvar}%
      \let\dtl@old=\DTLnegextent
      \DTLmin{\DTLnegextent}{\dtl@old}{\dtl@barvar}%
    }%
  }%
  \ifx\dtlbar@yticgap\relax
  \else
    \ifthenelse{\DTLisFPlt{\DTLnegextent}{0}}{%
    \edef\dtl@thistick{0}%
    \whiledo{\DTLisFPclosedbetween{\dtl@thistick}{\DTLnegextent}{0}}{%
      \dtlsub{\dtl@thistick}{\dtl@thistick}{\dtlbar@yticgap}%
    }%
    \let\DTLnegextent=\dtl@thistick
    }{}%
  \fi
\fi
```

Determine scaling factor

```
\@dtl@tmpcount=\DTLbarchartlength
\dtlsub{\dtl@extent}{\DTLbarmax}{\DTLnegextent}%
\dtldiv{\dtl@unit}{\number\@dtl@tmpcount}{\dtl@extent}%
```

Construct *y* tick list if required

```
 \setlength{\dtl@yticlabelwidth}{0pt}%
 \ifDTLbarytics
   \ifx\dtlbar@yticlist\relax
     \ifx\dtlbar@yticgap\relax
       \@dtl@tmpcount=\DTLmintickgap
       \divide\@dtl@tmpcount by 65536\relax
       \dtldiv{\dtl@mingap}{\number\@dtl@tmpcount}{\dtl@unit}%
       \dtl@constructticklist\DTLnegextent\DTLbarmax
         \dtl@mingap\dtlbar@yticlist
     \else
       \dtl@constructticklistwithgapz
         \DTLnegextent\DTLbarmax\dtlbar@yticlist\dtlbar@yticgap
     \fi
   \fi
   \ifx\dtlbar@ylabel\relax
   \else
     \ifx\dtlbar@yticlabels\relax
       \@for\dtl@thislabel:=\dtlbar@yticlist\do{%
         \dtlround{\dtl@thislabel}{\dtl@thislabel}
                 {\c@DTLbarroundvar}%
         \ifDTLverticalbars
           \settowidth{\dtl@tmplength}{%
```

```
                \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \else
             \settoheight{\dtl@tmplength}{%
                \DTLbardisplayYticklabel{\dtl@thislabel}}%
             \edef\@dtl@h{\the\dtl@tmplength}%
             \settodepth{\dtl@tmplength}{%
                \DTLbardisplayYticklabel{\dtl@thislabel}}%
             \addtolength{\dtl@tmplength}{\@dtl@h}%
             \addtolength{\dtl@tmplength}{\baselineskip}%
           \fi
           \ifdim\dtl@tmplength>\dtl@yticlabelwidth
             \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
           \fi
         }%
       \else
         \@for\dtl@thislabel:=\dtlbar@yticlabels\do{%
           \ifDTLverticalbars
             \settowidth{\dtl@tmplength}{%
               \DTLbardisplayYticklabel{\dtl@thislabel}}%
           \else
             \settoheight{\dtl@tmplength}{%
                \DTLbardisplayYticklabel{\dtl@thislabel}}%
             \edef\@dtl@h{\the\dtl@tmplength}%
             \settodepth{\dtl@tmplength}{%
                \DTLbardisplayYticklabel{\dtl@thislabel}}%
             \addtolength{\dtl@tmplength}{\@dtl@h}%
             \addtolength{\dtl@tmplength}{\baselineskip}%
           \fi
           \ifdim\dtl@tmplength>\dtl@yticlabelwidth
             \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
           \fi
         }%
       \fi
     \fi
   \fi
```

Calculate the offset for the lower label and number of labels

```
\dtl@xticlabelheight=0pt\relax
\@dtl@tmpcount=0\relax
\@for\dtl@thislabel:=\dtl@multibarlabels\do{%
  \advance\@dtl@tmpcount by 1\relax
  \settoheight{\dtl@tmplength}{\tikz\expandafter\pgftext\expandafter
    [\DTLbarXlabelalign]{\DTLdisplaylowerbarlabel{\dtl@thislabel}};}%
  \edef\@dtl@h{\the\dtl@tmplength}%
  \settodepth{\dtl@tmplength}{\tikz\expandafter\pgftext\expandafter
    [\DTLbarXlabelalign]{\DTLdisplaylowerbarlabel{\dtl@thislabel}};}%
  \addtolength{\dtl@tmplength}{\@dtl@h}%
  \addtolength{\dtl@tmplength}{\baselineskip}%
  \ifdim\dtl@tmplength>\dtl@xticlabelheight
    \setlength{\dtl@xticlabelheight}{\dtl@tmplength}%
  \fi
}
```

Calculate number of bars per group

```
\@dtl@tmpcount=0\relax
\@for\dtl@this:=\dtlbar@variables\do{%
  \advance\@dtl@tmpcount by 1\relax
}%
\edef\DTLbargroupwidth{\number\@dtl@tmpcount}%
```

Compute the total width of the bar chart (in terms of the x unit vector.)

```
\edef\dtl@n{\expandafter\number\csname dtlrows@#3\endcsname}
\dtlmul{\dtl@tmpi}{\dtl@n}{\DTLbargroupwidth}
\dtlsub{\dtl@tmpii}{\dtl@n}{1}%
\dtlmul{\dtl@tmpii}{\dtl@tmpii}{\dtlbar@groupgap}%
\dtladd{\DTLbarchartwidth}{\dtl@tmpi}{\dtl@tmpii}
```

Do the bar chart

```
\begin{tikzpicture}
```

Set unit vectors

```
\ifDTLverticalbars
  \pgfsetyvec{\pgfpoint{0pt}{\dtl@unit sp}}%
  \pgfsetxvec{\pgfpoint{\DTLbarwidth}{0pt}}%
\else
  \pgfsetxvec{\pgfpoint{\dtl@unit sp}{0pt}}%
  \pgfsetyvec{\pgfpoint{0pt}{\DTLbarwidth}}%
\fi
```

Begin hook

```
\DTLbaratbegintikz
```

Initialise

```
\def\@dtl@start{0}%
```

Iterate through data

```
\@sDTLforeach[#1]{#3}{#4}{%
```

Store the bar number in `\@dtl@bar`

```
\@dtl@barcount = 1\relax
```

Set the multibar label lists

```
\let\dtl@multibar@labels=\dtl@multibarlabels
\let\dtl@uppermultibar@labels=\dtl@uppermultibarlabels
```

Compute mid point over group

```
\dtlmul{\dtl@multimidpt}{\DTLbargroupwidth}{0.5}%
\dtladd{\dtl@multimidpt}{\dtl@multimidpt}{\@dtl@start}%
```

Iterate through each variable

```
\@for\DTLbarvariable:=\dtlbar@variables\do{%
```

Set end point

```
\dtladd{\@dtl@endpt}{\@dtl@start}{1}%
```

Convert variable to decimal

```
\expandafter\DTLconverttodecimal\expandafter
  {\DTLbarvariable}{\dtl@variable}%
```

Get the current lower label:

```
\dtl@chopfirst\dtl@multibar@labels\dtl@thisbarlabel\dtl@rest
\let\dtl@multibar@labels=\dtl@rest
```

Get the current upper label:

```
\dtl@chopfirst\dtl@uppermultibar@labels\dtl@thisupperbarlabel\dtl@rest
\let\dtl@uppermultibar@labels=\dtl@rest
```

Draw bars

```
\begin{scope}
 \expandafter\color\expandafter{\DTLgetbarcolor{\@dtl@barcount}}%
 \ifDTLverticalbars
   \fill (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@endpt,\dtl@variable) -- (\@dtl@endpt,0) -- cycle;
 \else
   \fill (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@endpt) -- (0,\@dtl@endpt) -- cycle;
 \fi
\end{scope}
```

Draw outline

```
\begin{scope}
\ifdim\DTLbaroutlinewidth>0pt
 \expandafter\color\expandafter{\DTLbaroutlinecolor}
 \ifDTLverticalbars
   \draw (\@dtl@start,0) -- (\@dtl@start,\dtl@variable)
      -- (\@dtl@endpt,\dtl@variable) -- (\@dtl@endpt,0) -- cycle;
 \else
   \draw (0,\@dtl@start) -- (\dtl@variable,\@dtl@start)
      -- (\dtl@variable,\@dtl@endpt) -- (0,\@dtl@endpt) -- cycle;
 \fi
\fi
\end{scope}
```

Calculate mid point

```
\dtladd{\@dtl@midpt}{\@dtl@start}{0.5}%
```

Draw lower *x* labels

```
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\@dtl@midpt}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}},
     \DTLbarXlabelalign
  }%
  \edef\DTLstartpt{\noexpand\pgfpointxy{\@dtl@midpt}{0}}%
\else
  \edef\dtl@textopt{%
```

```
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{0}{\@dtl@midpt}}
             {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}},
       \DTLbarXlabelalign
    }%
    \edef\DTLstartpt{\noexpand\pgfpointxy{0}{\@dtl@midpt}}%
  \fi
   \expandafter\pgftext\expandafter[\dtl@textopt]{%
     \DTLdisplaylowermultibarlabel{\dtl@thisbarlabel}}
```

Draw upper *x* labels

```
  \ifDTLverticalbars
    \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}
    {
      \edef\dtl@textopt{%
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}
             {\noexpand\pgfpoint{0pt}{-\noexpand\DTLbarlabeloffset}}}
      }%
    }{%
      \edef\dtl@textopt{%
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}
             {\noexpand\pgfpoint{0pt}{\noexpand\DTLbarlabeloffset}}}
      }%
    }
    \edef\DTLendpt{\noexpand\pgfpointxy{\@dtl@midpt}{\dtl@variable}}%
  \else
    \expandafter\DTLifnumlt\expandafter{\DTLbarvariable}{0}
    {
      \edef\dtl@textopt{right,
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}
             {\noexpand\pgfpoint{-\noexpand\DTLbarlabeloffset}{0pt}}}
      }%
    }{%
      \edef\dtl@textopt{left,
        at={\noexpand\pgfpointadd
             {\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}
             {\noexpand\pgfpoint{\noexpand\DTLbarlabeloffset}{0pt}}}
      }%
    }
    \edef\DTLendpt{\noexpand\pgfpointxy{\dtl@variable}{\@dtl@midpt}}%
  \fi
   \expandafter\pgftext\expandafter[\dtl@textopt]{%
      \DTLdisplayuppermultibarlabel{\dtl@thisupperbarlabel}}
```

Set the mid point

```
  \def\DTLmidpt{\pgfpointlineattime{0.5}{\DTLstartpt}{\DTLendpt}}%
```

Do every bar hook

```
\DTLeverybarhook
```

End of loop increment loop variables

```
  \dtladd{\@dtl@start}{\@dtl@start}{1}%
  \advance\@dtl@barcount by 1\relax
}%
% Draw lower group $x$ labels
%    \begin{macrocode}
\setlength{\dtl@tmplength}{\DTLbarlabeloffset}%
\addtolength{\dtl@tmplength}{\dtl@xticlabelheight}%
\ifDTLverticalbars
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{\dtl@multimidpt}{0}}
           {\noexpand\pgfpoint{0pt}{-\noexpand\dtl@tmplength}}},
     \DTLbarXlabelalign
  }%
\else
  \edef\dtl@textopt{%
      at={\noexpand\pgfpointadd
           {\noexpand\pgfpointxy{0}{\dtl@multimidpt}}
           {\noexpand\pgfpoint{-\noexpand\dtl@tmplength}{0pt}}},
     \DTLbarXlabelalign
  }%
\fi
 \expandafter\pgftext\expandafter[\dtl@textopt]{%
   \DTLdisplaylowerbarlabel{\dtl@barlabel}}
```

Increment starting position by \dtlbar@groupgap

```
  \dtladd{\@dtl@start}{\@dtl@start}{\dtlbar@groupgap}%
}
```

Draw *x* axis

```
\ifDTLbarxaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (\DTLbarchartwidth,0);
  \else
    \expandafter\draw\expandafter[\DTLBarXAxisStyle]
      (0,0) -- (0,\DTLbarchartwidth);
  \fi
\fi
```

Draw *y* axis

```
\ifDTLbaryaxis
  \ifDTLverticalbars
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (0,\DTLnegextent) -- (0,\DTLbarmax);
  \else
    \expandafter\draw\expandafter[\DTLBarYAxisStyle]
      (\DTLnegextent,0) -- (\DTLbarmax,0);
    \fi
  \fi
```

Plot *y* tick marks if required

```
  \ifx\dtlbar@yticlist\relax
  \else
    \@for\dtl@thistick:=\dtlbar@yticlist\do{%
      \ifDTLverticalbars
        \pgfpathmoveto{\pgfpointxy{0}{\dtl@thistick}}
        \pgfpathlineto{
          \pgfpointadd{\pgfpointxy{0}{\dtl@thistick}}
                      {\pgfpoint{-\DTLticklength}{0pt}}}
      \else
        \pgfpathmoveto{\pgfpointxy{\dtl@thistick}{0}}
        \pgfpathlineto{
          \pgfpointadd{\pgfpointxy{\dtl@thistick}{0}}
                      {\pgfpoint{0pt}{-\DTLticklength}}}
      \fi
      \pgfusepath{stroke}
      \ifx\dtlbar@yticlabels\relax
         \dtlround{\dtl@thislabel}{\dtl@thistick}
                  {\c@DTLbarroundvar}%
      \else
         \dtl@chopfirst\dtlbar@yticlabels\dtl@thislabel\dtl@rest
         \let\dtlbar@yticlabels=\dtl@rest
      \fi
      \ifDTLverticalbars
        \edef\dtl@textopt{\DTLbarYticklabelalign,%
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{0}{\dtl@thistick}}
               {\noexpand\pgfpoint{-\noexpand\DTLticklabeloffset}{0pt}},
          }}%
      \else
        \edef\dtl@textopt{\DTLbarYticklabelalign,
          at={\noexpand\pgfpointadd
               {\noexpand\pgfpointxy{\dtl@thistick}{0}}
               {\noexpand\pgfpoint{0pt}{-\noexpand\DTLticklabeloffset}}
          }}%
      \fi
      \expandafter\pgftext\expandafter[\dtl@textopt]{%
        \DTLbardisplayYticklabel{\dtl@thislabel}}
    }%
  \fi
```

Plot the *y* label if required

```
  \ifx\dtlbar@ylabel\relax
  \else
    \addtolength{\dtl@yticlabelwidth}{\baselineskip}%
    \setlength{\dtl@tmplength}{0.5\DTLbarchartlength}
    \ifDTLverticalbars
```

```
      \pgftext[bottom,center,at={\pgfpointadd
          {\pgfpointxy{0}{\DTLnegextent}}%
          {\pgfpoint{-\dtl@yticlabelwidth}{\dtl@tmplength}}},
          rotate=90]{%
        \dtlbar@ylabel}
    \else
      \pgftext[bottom,center,at={\pgfpointadd
          {\pgfpointxy{\DTLnegextent}{0}}%
          {\pgfpoint{\dtl@tmplength}{-\dtl@yticlabelwidth}}}]{%
        \dtlbar@ylabel}
    \fi
  \fi
```

Finish bar chart

```
  \DTLbaratendtikz
  \end{tikzpicture}
  \fi
  }}
```

8 datapie.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datapie}[2019/09/27 v2.32 (NLCT)]
```

Require xkeyval package

```
\RequirePackage{xkeyval}
```

TLcolorpiechart

The conditional `\ifDTLcolorpiechart` is to determine whether to use colour or grey scale.

```
\newif\ifDTLcolorpiechart
\DTLcolorpiecharttrue
```

Package options to change the conditional:

```
\DeclareOption{color}{\DTLcolorpiecharttrue}
\DeclareOption{gray}{\DTLcolorpiechartfalse}
```

fDTLrotateinner

Define boolean keys to govern label rotations.

```
\define@boolkey{datapie}[DTL]{rotateinner}[true]{}
```

fDTLrotateouter

```
\define@boolkey{datapie}[DTL]{rotateouter}[true]{}
```

Set defaults:

```
\DTLrotateinnerfalse
\DTLrotateouterfalse
```

Package options to change `\DTLrotateinner`

```
\DeclareOption{rotateinner}{\DTLrotateinnertrue}
\DeclareOption{norotateinner}{\DTLrotateinnerfalse}
```

Package options to change `\DTLrotateouter`

```
\DeclareOption{rotateouter}{\DTLrotateoutertrue}
\DeclareOption{norotateouter}{\DTLrotateouterfalse}
```

Process options:

```
\ProcessOptions
```

Required packages:

```
\RequirePackage{datatool}
\RequirePackage{tikz}
```

Define some variables that govern the appearance of the pie chart.

\DTLradius The radius of the pie chart is given by \DTLradius.

```
\newlength\DTLradius
\DTLradius=2cm
```

\DTLinnerratio The inner label offset ratio is given by \DTLinnerratio

```
\newcommand*{\DTLinnerratio}{0.5}
```

\DTLouterratio The outer label offset ratio is given by \DTLouterratio.

```
\newcommand*{\DTLouterratio}{1.25}
```

DTLcutawayratio The cutaway offset ratio is given by \DTLcutawayratio.

```
\newcommand*\DTLcutawayratio{0.2}
```

\DTLstartangle The angle of the first segment is given by \DTLstartangle.

```
\newcommand*{\DTLstartangle}{0}
```

dtl@inneroffset

```
\newlength\dtl@inneroffset
\dtl@inneroffset=\DTLinnerratio\DTLradius
```

dtl@outeroffset

```
\newlength\dtl@outeroffset
\dtl@outeroffset=\DTLouterratio\DTLradius
```

l@cutawayoffset

```
\newlength\dtl@cutawayoffset
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius
```

dtl@piecutaways \dtl@piecutaways is a comma separated list of segments that need to be cut away from the pie chart.

```
\newcommand*{\dtl@piecutaways}{}
```

\dtl@innerlabel \dtl@innerlabel specifies the label to appear inside the segment. By default this is the variable used to create the pie chart.

```
\def\dtl@innerlabel{\DTLpievariable}%
```

\dtl@outerlabel

```
\def\dtl@outerlabel{}%
```

DTLpieroundvar DTLpieroundvar is a counter governing the number of digits to round to when using \FPround.

```
\newcounter{DTLpieroundvar}
\setcounter{DTLpieroundvar}{1}
```

isplayinnerlabel

\DTLdisplayinnerlabel{⟨*label*⟩}

This is used to format the inner label. This just does the label by default.

```
\newcommand*{\DTLdisplayinnerlabel}[1]{#1}
```

isplayouterlabel

```
\DTLdisplayouterlabel{⟨label⟩}
```

This is used to format the outer label. This just does the label by default.

```
\newcommand*{\DTLdisplayouterlabel}[1]{#1}
```

\DTLpiepercent

\DTLpiepercent returns the percentage value of the current segment.

```
\newcommand*{\DTLpiepercent}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{datapie}{Can't use
  \string\DTLpiepercent\space outside
  \string\DTLpiechart}{}%
\else
  \csname dtl@piepercent@\romannumeral\@dtl@seg\endcsname
\fi}
```

Lpieatbegintikz

\DTLpieatbegintikz specifies any commands to apply at the start of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLpieatbegintikz}{}
```

DTLpieatendtikz

\DTLpieatendtikz specifies any commands to apply at the end of the tikzpicture environment. By default it does nothing.

```
\newcommand*{\DTLpieatendtikz}{}
```

tpiesegmentcolor

```
\DTLsetpiesegmentcolor{⟨n⟩}{⟨color⟩}
```

Assign colour name ⟨*color*⟩ to the ⟨*n*⟩th segment.

```
\newcommand*{\DTLsetpiesegmentcolor}[2]{%
\expandafter\def\csname dtlpie@segcol\romannumeral#1\endcsname{#2}%
}
```

tpiesegmentcolor

```
\DTLgetpiesegmentcolor{⟨n⟩}
```

Get the colour specification for segment ⟨*n*⟩

```
\newcommand*{\DTLgetpiesegmentcolor}[1]{%
\csname dtlpie@segcol\romannumeral#1\endcsname}
```

opiesegmentcolor

```
\DTLdopiesegmentcolor{⟨n⟩}
```

Set the colour to that for segment ⟨*n*⟩

```
\newcommand*{\DTLdopiesegmentcolor}[1]{%
\expandafter\color\expandafter
{\csname dtlpie@segcol\romannumeral#1\endcsname}}
```

piesegmentcolor \DTLdocurrentpiesegmentcolor sets the colour to that of the current segment.

```
\newcommand*{\DTLdocurrentpiesegmentcolor}{%
\ifnum\dtlforeachlevel=0\relax
  \PackageError{datapie}{Can't use
  \string\DTLdocurrentpiesegmentcolor\space outside
  \string\DTLpiechart}{}%
\else
  \expandafter\DTLdopiesegmentcolor\expandafter{%
  \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
\fi}
```

pieoutlinecolor \DTLpieoutlinecolor specifies what colour to draw the outline.

```
\newcommand*{\DTLpieoutlinecolor}{black}
```

pieoutlinewidth \DTLpieoutlinewidth specifies the line width of the outline: Outline is only drawn if the linewidth is greater than 0pt.

```
\newlength\DTLpieoutlinewidth
\DTLpieoutlinewidth=0pt
```

Set the default colours. If there are more than eight segments, more colours will need to be defined.

```
\ifDTLcolorpiechart
 \DTLsetpiesegmentcolor{1}{red}
 \DTLsetpiesegmentcolor{2}{green}
 \DTLsetpiesegmentcolor{3}{blue}
 \DTLsetpiesegmentcolor{4}{yellow}
 \DTLsetpiesegmentcolor{5}{magenta}
 \DTLsetpiesegmentcolor{6}{cyan}
 \DTLsetpiesegmentcolor{7}{orange}
 \DTLsetpiesegmentcolor{8}{white}
\else
 \DTLsetpiesegmentcolor{1}{black!15}
 \DTLsetpiesegmentcolor{2}{black!25}
 \DTLsetpiesegmentcolor{3}{black!35}
 \DTLsetpiesegmentcolor{4}{black!45}
 \DTLsetpiesegmentcolor{5}{black!55}
 \DTLsetpiesegmentcolor{6}{black!65}
 \DTLsetpiesegmentcolor{7}{black!75}
 \DTLsetpiesegmentcolor{8}{black!85}
\fi
```

Define keys for \DTLpiechart optional argument. Set the starting angle of the first segment.

```
\define@key{datapie}{start}{\def\DTLstartangle{#1}}
```

Set the radius of the pie chart (must be set prior to inneroffset and outeroffset keys.)

```
\define@key{datapie}{radius}{\DTLradius=#1\relax
\dtl@inneroffset=\DTLinnerratio\DTLradius
\dtl@outeroffset=\DTLouterratio\DTLradius
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius}
```

Set the inner ratio.

```
\define@key{datapie}{innerratio}{%
\def\DTLinnerratio{#1}%
\dtl@inneroffset=\DTLinnerratio\DTLradius}
```

Set the outer ratio

```
\define@key{datapie}{outerratio}{%
\def\DTLouterratio{#1}%
\dtl@outeroffset=\DTLouterratio\DTLradius}
```

The cutaway offset ratio

```
\define@key{datapie}{cutawayratio}{%
\def\DTLcutawayratio{#1}%
\dtl@cutawayoffset=\DTLcutawayratio\DTLradius}
```

Set the inner offset as an absolute value (not dependent on the radius.)

```
\define@key{datapie}{inneroffset}{%
\dtl@inneroffset=#1}
```

Set the outer offset as an absolute value (not dependent on the radius.)

```
\define@key{datapie}{outeroffset}{%
\dtl@outeroffset=#1}
```

Set the cutaway offset as an absolute value (not dependent on the radius.)

```
\define@key{datapie}{cutawayoffset}{%
\dtl@cutawayoffset=#1}
```

List of cut away segments.

```
\define@key{datapie}{cutaway}{%
\renewcommand*{\dtl@piecutaways}{#1}}
```

Variable used to create the pie chart. (Must be a control sequence.)

```
\define@key{datapie}{variable}{%
\def\DTLpievariable{#1}}
```

Inner label

```
\define@key{datapie}{innerlabel}{%
\def\dtl@innerlabel{#1}}
```

Outer label

```
\define@key{datapie}{outerlabel}{%
\def\dtl@outerlabel{#1}}
```

\DTLpiechart

```
\DTLpiechart[⟨conditions⟩]{⟨option list⟩}{⟨db name⟩}{⟨assign list⟩}
```

Make a pie chart from data given in data base ⟨*db name*⟩, where ⟨*assign list*⟩ is a comma-separated list of ⟨*cmd*⟩=⟨*key*⟩ pairs. ⟨*option list*⟩ must include `variable=`⟨*cmd*⟩, where ⟨*cmd*⟩ is included in ⟨*assign list*⟩. The optional argument ⟨*conditions*⟩ is the same as that for `\DTLforeach`.

```
\newcommand*{\DTLpiechart}[4][\boolean{true}]{%
\bgroup
  \let\DTLpievariable=\relax
  \setkeys{datapie}{#2}%
  \ifx\DTLpievariable\relax
    \PackageError{datapie}%
    {\string\DTLpiechart\space missing variable}{}%
  \else
```

Compute the total.

```
    \def\dtl@total{0}%
    \@sDTLforeach[#1]{#3}{#4}{%
      \let\dtl@oldtotal=\dtl@total
      \expandafter\DTLconverttodecimal\expandafter
        {\DTLpievariable}{\dtl@variable}%
      \dtladd{\dtl@total}{\dtl@variable}{\dtl@total}%
    }%
```

Compute the angles

```
    \expandafter\DTLconverttodecimal\expandafter
      {\DTLstartangle}{\@dtl@start}%
    \@sDTLforeach[#1]{#3}{#4}{%
      \expandafter\DTLconverttodecimal\expandafter
        {\DTLpievariable}{\dtl@variable}%
      \dtl@computeangles
        {\csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname}%
        {\dtl@variable}%
      \expandafter\@dtl@seg\expandafter=
        \csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
      \dtlmul{\dtl@tmp}{\dtl@variable}{100}%
      \let\dtl@old=\dtl@tmp
      \dtldiv{\dtl@tmp}{\dtl@old}{\dtl@total}%
      \expandafter\dtlround
        \csname dtl@piepercent@\romannumeral\@dtl@seg\endcsname\dtl@tmp
        \c@DTLpieroundvar
    }%
```

Compute the offsets for each cut away segment

```
    \@for\dtl@row:=\dtl@piecutaways\do{%
      \expandafter\@dtl@set@off\dtl@row-\relax
    }%
```

Set the starting angle

```
    \let\dtl@start=\DTLstartangle
```

Do the pie chart

```
    \begin{tikzpicture}
```

```
\DTLpieatbegintikz
\@sDTLforeach[#1]{#3}{#4}%
{%
```

Store the segment number in `\@dtl@seg`

```
\expandafter\@dtl@seg\expandafter=
\csname c@DTLrow\romannumeral\dtlforeachlevel\endcsname%
```

Set the start angle.

```
\edef\dtl@start{%
  \csname dtl@sang@\romannumeral\@dtl@seg\endcsname}%
```

Set the extent

```
\edef\dtl@extent{%
  \csname dtl@angle@\romannumeral\@dtl@seg\endcsname}%
```

Compute the end angle

```
\dtladd{\dtl@endangle}{\dtl@start}{\dtl@extent}%
```

Compute the shift.

```
\edef\dtl@angle{%
   \csname dtl@cut@angle@\romannumeral\@dtl@seg\endcsname}%
\let\dtl@old=\dtl@angle
\dtl@truncatedecimal\dtl@angle
\ifnum\dtl@angle>180\relax
  \dtlsub{\dtl@angle}{\dtl@old}{360}%
  \dtl@truncatedecimal\dtl@angle
\fi
\edef\dtl@cutlen{%
  \csname dtl@cut@len@\romannumeral\@dtl@seg\endcsname
}%
\edef\@dtl@shift{(\dtl@angle:\dtl@cutlen)}%
```

Compute the mid way angle.

```
\dtlmul{\dtl@angle}{\dtl@extent}{0.5}%
\dtladd{\dtl@midangle}{\dtl@angle}{\dtl@start}%
```

Draw the segment.

```
\begin{scope}[shift={\@dtl@shift}]%

\fill[color=\DTLgetpiesegmentcolor\@dtl@seg] (0,0) --
(\dtl@start:\DTLradius)
arc (\dtl@start:\dtl@endangle:\DTLradius) -- cycle;
```

Draw the outline if required:

```
\ifdim\DTLpieoutlinewidth>0pt\relax
  \draw[color=\DTLpieoutlinecolor,%
        line width=\DTLpieoutlinewidth]
  (0,0) -- (\dtl@start:\DTLradius)
  arc (\dtl@start:\dtl@endangle:\DTLradius) -- cycle;
\fi
```

Convert decimal to an integer

```
\dtl@truncatedecimal\dtl@midangle
```

Determine whether to rotate inner labels

```
\ifDTLrotateinner
```

If the mid way angle is between 90 and 270, the text will look upside-down, so adjust accordingly.

```
  \dtlifnumopenbetween{\dtl@midangle}{90}{270}%
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
     \dtlifnumlt{\dtl@midangle}{-90}%
     {\let\@dtl@next\@firstoftwo}%
     {\let\@dtl@next\@secondoftwo}%
  }%
  \@dtl@next
  {%
    \dtlsub{\dtl@labelangle}{\dtl@midangle}{180}%
    \dtl@truncatedecimal\dtl@labelangle
    \edef\dtl@innernodeopt{anchor=east,rotate=\dtl@labelangle}%
  }%
  {%
    \edef\dtl@innernodeopt{anchor=west,rotate=\dtl@midangle}%
  }%
```

Don't rotate inner labels

```
\else
  \edef\dtl@innernodeopt{anchor=center}%
\fi
```

Determine whether to rotate outer labels

```
\ifDTLrotateouter
```

If the mid way angle is between 90 and 270, the text will look upside-down, so adjust accordingly.

```
  \dtlifnumopenbetween{\dtl@midangle}{90}{270}%
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
     \dtlifnumlt{\dtl@midangle}{-90}%
     {\let\@dtl@next\@firstoftwo}%
     {\let\@dtl@next\@secondoftwo}%
  }%
  \@dtl@next
  {%
    \dtlsub{\dtl@labelangle}{\dtl@midangle}{180}%
    \dtl@truncatedecimal\dtl@labelangle
    \edef\dtl@outernodeopt{anchor=east,rotate=\dtl@labelangle}%
```

```
 }%
 {%
   \edef\dtl@outernodeopt{anchor=west,rotate=\dtl@midangle}%
 }%
```

Don't rotate outer labels

```
\else
```

If ($\theta > -45$ and $\theta < 45$) or $\theta = 45$ or $\theta > 315$

```
\dtlifnumeq{\dtl@midangle}{45}
{%
  \let\@dtl@next\@firstoftwo
}%
{%
  \dtlifnumgt{\dtl@midangle}{315}
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
    \dtlifnumopenbetween{\dtl@midangle}{-45}{45}%
    {%
      \let\@dtl@next\@firstoftwo
    }%
    {%
      \let\@dtl@next\@secondoftwo
    }%
  }%
}%
\@dtl@next
{%
```

East quadrant

```
  \edef\dtl@outernodeopt{anchor=west}%
}%
{%
  \dtlifnumopenbetween{\dtl@midangle}{45}{135}%
  {%
    \let\@dtl@next\@firstoftwo
  }%
  {%
    \dtlifnumeq{\dtl@midangle}{135}%
    {%
      \let\@dtl@next\@firstoftwo
    }%
    {%
      \let\@dtl@next\@secondoftwo
    }%
  }%
  \@dtl@next
  {%
```

North quadrant

```
      \edef\dtl@outernodeopt{anchor=south}%
    }%
    {%
```

If ($\theta > 135$ and $\theta < 225$) or $\theta = 225$ or $\theta = -135$ or $\theta < -135$

```
      \dtlifnumopenbetween{\dtl@midangle}{135}{225}%
      {%
        \let\@dtl@next\@firstoftwo
      }%
      {%
        \dtlifnumeq{\dtl@midangle}{225}%
        {%
          \let\@dtl@next\@firstoftwo
        }%
        {%
          \dtlifnumeq{\dtl@midangle}{-135}%
          {%
            \let\@dtl@next\@firstoftwo
          }%
          {%
            \dtlifnumlt{\dtl@midangle}{-135}%
            {%
              \let\@dtl@next\@firstoftwo
            }%
            {%
              \let\@dtl@next\@secondoftwo
            }%
          }%
        }%
      }%
      \@dtl@next
      {%
```

West quadrant

```
        \edef\dtl@outernodeopt{anchor=east}%
      }%
      {%
        \edef\dtl@outernodeopt{anchor=north}%
      }%
    }%
  }%
\fi
```

Draw inner and outer labels

```
\edef\@dtl@dolabel{%
  \noexpand\draw (\dtl@midangle:\the\dtl@inneroffset)
     node[\dtl@innernodeopt]{%
   \noexpand\DTLdisplayinnerlabel{\noexpand\dtl@innerlabel}};
}%
```

```
        \@dtl@dolabel
        \edef\@dtl@dolabel{%
          \noexpand\draw (\dtl@midangle:\the\dtl@outeroffset)
            node[\dtl@outernodeopt]{%
          \noexpand\DTLdisplayouterlabel{\noexpand\dtl@outerlabel}};
        }%
        \@dtl@dolabel
        \end{scope}%
      }%
      \DTLpieatendtikz
    \end{tikzpicture}%
  \fi
\egroup
}
```

tl@computeangles

\dtl@computeangles{⟨*n*⟩}{⟨*variable*⟩}

Compute the angles for segment ⟨*n*⟩. This sets \dtl@sang@⟨*n*⟩ (start angle), \dtl@angle@⟨*n*⟩ (extent angle), \dtl@cut@angle@⟨*n*⟩ (cut away angle) and \dtl@cut@len@⟨*n*⟩ (cut away length).

```
\newcommand*{\dtl@computeangles}[2]{%
  \dtlifnumgt{\@dtl@start}{180}%
  {%
```

if startangle > 180

```
    \let\dtl@old=\@dtl@start
```

startangle := startangle $- 360$

```
    \dtlsub{\@dtl@start}{\dtl@old}{360}%
  }%
  {}%
  \dtlifnumlt{\@dtl@start}{-180}%
  {%
```

if startangle < -180

```
    \let\dtl@old=\@dtl@start
```

startangle = startangle $+ 360$

```
    \dtladd{\@dtl@start}{\dtl@old}{360}%
  }%
  {}%
  \expandafter\edef\csname dtl@sang@\romannumeral#1\endcsname{%
     \@dtl@start}%
  \dtlmul{\dtl@angle}{360}{#2}%
  \let\dtl@old=\dtl@angle
  \dtldiv{\dtl@angle}{\dtl@old}{\dtl@total}%
  \expandafter\let\csname dtl@angle@\romannumeral#1\endcsname=\dtl@angle
  \let\dtl@old=\@dtl@start
  \dtladd{\@dtl@start}{\dtl@old}{\dtl@angle}%
  \expandafter\def\csname dtl@cut@angle@\romannumeral#1\endcsname{0}%
  \expandafter\def\csname dtl@cut@len@\romannumeral#1\endcsname{0cm}%
}
```

Set the offset angles.

\@dtl@set@off

```
\def\@dtl@set@off#1-#2\relax{%
  \ifstrempty{#2}%
  {%
    \@@dtl@set@off{#1}%
  }%
  {%
    \@@dtl@set@offr#1-#2\relax
  }%
}
```

Set offset for individual segment:

\@@dtl@set@off

```
\newcommand*{\@@dtl@set@off}[1]{%
  \edef\dtl@old{\csname dtl@angle@\romannumeral#1\endcsname}%
  \dtlmul{\dtl@angle}{\dtl@old}{0.5}%
  \let\dtl@old=\dtl@angle
  \edef\dtl@sang{\csname dtl@sang@\romannumeral#1\endcsname}%
  \dtladd{\dtl@angle}{\dtl@old}{\dtl@sang}%
  \expandafter\edef\csname dtl@cut@angle@\romannumeral#1\endcsname{%
  \dtl@angle}%
  \expandafter\edef\csname dtl@cut@len@\romannumeral#1\endcsname{%
   \the\dtl@cutawayoffset}%
}
```

Define count register to keep track of segments

\@dtl@seg

```
\newcount\@dtl@seg
```

\@@dtl@setoffr Set offset for a range of segments

```
\def\@@dtl@set@offr#1-#2-\relax{%
 \ifnum#1>#2\relax
   \PackageError{datapie}{Segment ranges must go in ascending order}{%
   Try #2-#1 instead of #1-#2}%
 \else
   \def\dtl@angle{0}%
   \@dtl@seg=#1\relax
   \whiledo{\not\(\@dtl@seg > #2\)}{%
    \let\dtl@old=\dtl@angle
    \edef\dtl@segang{\csname dtl@angle@\romannumeral\@dtl@seg\endcsname}%
    \dtladd{\dtl@angle}{\dtl@old}{\dtl@segang}%
    \advance\@dtl@seg by 1\relax
   }%
   \let\dtl@old=\dtl@angle
   \dtlmul{\dtl@angle}{\dtl@old}{0.5}%
   \edef\dtl@sang{\csname dtl@sang@\romannumeral#1\endcsname}%
   \let\dtl@old=\dtl@angle
   \dtladd{\dtl@angle}{\dtl@old}{\dtl@sang}%
   \@dtl@seg=#1\relax
   \whiledo{\not\(\@dtl@seg > #2\)}{%
   \expandafter
    \let\csname dtl@cut@angle@\romannumeral\@dtl@seg\endcsname
    =\dtl@angle
   \expandafter
    \edef\csname dtl@cut@len@\romannumeral\@dtl@seg\endcsname{%
    \the\dtl@cutawayoffset}
    \advance\@dtl@seg by 1\relax
   }%
 \fi
}
```

9 dataplot.sty

Declare package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{dataplot}[2019/09/27 v2.32 (NLCT)]
```

Required packages

```
\RequirePackage{xkeyval}
\RequirePackage{tikz}
\RequirePackage{datatool}
```

Load TikZ plot libraries

```
\usetikzlibrary{plotmarks}
\usetikzlibrary{plothandlers}
```

Load calc library

```
\usetikzlibrary{calc}
```

\DTLplotstream

```
\DTLplotstream[⟨condition⟩]{⟨db name⟩}{⟨x key⟩}{⟨y key⟩}
```

Add points to a stream from the database called ⟨*db name*⟩ where the *x* co-ordinates are given by the key ⟨*x key*⟩ and the *y* co-ordinates are given by the key ⟨*y key*⟩. The optional argument ⟨*condition*⟩ is the same as that for `\DTLforeach`

```
\newcommand*{\DTLplotstream}[4][\boolean{true}]{%
  \@sDTLforeach[#1]{#2}{\dtl@x=#3,\dtl@y=#4}{%
    \DTLconverttodecimal{\dtl@x}{\dtl@decx}%
    \DTLconverttodecimal{\dtl@y}{\dtl@decy}%
    \pgfplotstreampoint{\pgfpointxy{\dtl@decx}{\dtl@decy}}%
  }%
}
```

\DTLplotmarks `\DTLplotmarks` contains a list of plot marks used by `\DTLplot`.

```
\newcommand*{\DTLplotmarks}{%
  \pgfuseplotmark{o},%
  \pgfuseplotmark{x},%
  \pgfuseplotmark{+},%
  \pgfuseplotmark{square},%
  \pgfuseplotmark{triangle},%
  \pgfuseplotmark{diamond},%
  \pgfuseplotmark{pentagon},%
  \pgfuseplotmark{asterisk},%
  \pgfuseplotmark{star}%
}
```

Lplotmarkcolors \DTLplotmarkcolors contains a list of the plot mark colours.

```
\newcommand*{\DTLplotmarkcolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
  cyan,%
  orange,%
  black,%
  gray}
```

\DTLplotlines \DTLplotlines contains a list of dash patterns used by \DLTplot.

```
\newcommand*{\DTLplotlines}{%
  \pgfsetdash{}{0pt},% solid line
  \pgfsetdash{{10pt}{5pt}}{0pt},%
  \pgfsetdash{{5pt}{5pt}}{0pt},%
  \pgfsetdash{{1pt}{5pt}}{0pt},%
  \pgfsetdash{{5pt}{5pt}{1pt}{5pt}}{0pt},%
  \pgfsetdash{{1pt}{3pt}}{0pt},%
}
```

Lplotlinecolors \DTLplotlinecolors contains a list of the plot line colours.

```
\newcommand*{\DTLplotlinecolors}{%
  red,%
  green,%
  blue,%
  yellow,%
  magenta,%
  cyan,%
  orange,%
  black,%
  gray}
```

\DTLplotwidth The default total plot width is stored in the length \dtlplotwidth

```
\newlength\DTLplotwidth
\setlength\DTLplotwidth{4in}
```

\DTLplotheight The default total plot height is stored in the length \dtlplotheight

```
\newlength\DTLplotheight
\setlength\DTLplotheight{4in}
```

\DTLticklength The length of the tick marks is given by \DTLticklength

```
\newlength\DTLticklength
\setlength\DTLticklength{5pt}
```

minorticklength The length of the minor tick marks is given by \DTLminorticklength.

```
\newlength\DTLminorticklength
\setlength\DTLminorticklength{2pt}
```

ticklabeloffset The offset from the axis to the tick label is given by \DTLticklabeloffset.

```
\newlength\DTLticklabeloffset
\setlength\DTLticklabeloffset{8pt}
```

xticlabelheight \dtl@xticlabelheight is used to store the height of the *x* tick labels.

```
\newlength\dtl@xticlabelheight
```

@yticlabelwidth \dtl@yticlabelwidth is used to store the width of the *y* tick labels.

```
\newlength\dtl@yticlabelwidth
```

\DTLmintickgap \DTLmintickgap stores the suggested minimum distance between tick marks where the gap is not specified.

```
\newlength\DTLmintickgap
\setlength\DTLmintickgap{20pt}
```

minminortickgap The suggested minimum distance between minor tick marks where the gap is not specified is given by \DTLminminortickgap.

```
\newlength\DTLminminortickgap
\setlength\DTLminminortickgap{5pt}
```

DTLplotroundvar Round *x* tick labels to the number of digits given by the counter DTLplotroundXvar.

```
\newcounter{DTLplotroundXvar}
\setcounter{DTLplotroundXvar}{2}
```

TLplotroundYvar Round *y* tick labels to the number of digits given by the counter DTLplotroundYvar.

```
\newcounter{DTLplotroundYvar}
\setcounter{DTLplotroundYvar}{2}
```

\ifDTLxaxis The conditional \ifDTLxaxis is used to determine whether or not to display the *x* axis.

```
\newif\ifDTLxaxis
\DTLxaxistrue
```

\DTLXAxisStyle The style of the *x* axis is given by \DTLXAxisStyle. This is just a solid line by default.

```
\newcommand*{\DTLXAxisStyle}{-}
```

\ifDTLyaxis The conditional \ifDTLyaxis is used to determine whether or not to display the *y* axis

```
\newif\ifDTLyaxis
\DTLyaxistrue
```

\DTLYAxisStyle The style of the *y* axis is given by \DTLYAxisStyle. This is just a solid line by default.

```
\newcommand*{\DTLYAxisStyle}{-}
```

Lmajorgridstyle The style of the major grid lines is given by \DTLmajorgridstyle.

```
\newcommand*{\DTLmajorgridstyle}{color=gray,-}
```

Lminorgridstyle The style of the minor grid lines is given by \DTLminorgridstyle.

```
\newcommand*{\DTLminorgridstyle}{color=gray,loosely dotted}
```

\ifDTLxticsin The conditional \ifDTLxticsin is used to determine whether the *x* tics should point in or out.

```
\newif\ifDTLxticsin
\DTLxticsintrue
```

\ifDTLyticsin The conditional \ifDTLyticsin is used to determine whether the *y* tics should point in or out.

```
\newif\ifDTLyticsin
\DTLyticsintrue
```

l@legendsetting The legend setting is stored in the count register \dtl@legendsetting.

```
\newcount\dtl@legendsetting
```

TLlegendxoffset The gap between the border of plot and legend is given by the lengths \DTLlegendxoffset and \DTLlegendyoffset

```
\newlength\DTLlegendxoffset
\setlength\DTLlegendxoffset{10pt}
```

TLlegendyoffset

```
\newlength\DTLlegendyoffset
\setlength\DTLlegendyoffset{10pt}
```

\DTLformatlegend

\DTLformatlegend{⟨*legend*⟩}

This formats the legend.

```
\newcommand*{\DTLformatlegend}[1]{%
\setlength{\fboxrule}{1.1pt}%
\fcolorbox{black}{white}{#1}}
```

fDTLshowmarkers The conditional \ifDTLshowmarkers is used to specify whether or not to use markers.

```
\newif\ifDTLshowmarkers
\DTLshowmarkerstrue
```

\ifDTLshowlines The conditional \ifDTLshowlines is used to specify whether or not to use lines.

```
\newif\ifDTLshowlines
\DTLshowlinesfalse
```

plotatbegintikz \DTLplotatbegintikz is a hook to insert stuff at the start of the tikzpicture environment (after the unit vectors have been set).

```
\newcommand*{\DTLplotatbegintikz}{}
```

plothandlermark Provide a convenient access to \pgfplothandlermark that reverses the effect of the plot scaling.

```
\newcommand*{\@dtlplothandlermark}[1]{%
  \pgfplothandlermark
  {%
    \pgfmathparse{1/\dtl@scale@x}%
    \pgftransformxscale{\pgfmathresult}%
    \pgfmathparse{1/\dtl@scale@y}%
    \pgftransformyscale{\pgfmathresult}%
   #1%
   }%
}
```

Just in case a user attempts to use \dtlplothandermark outside \DTLplot:

```
\newcommand*{\dtlplothandlermark}[1]{%
   \PackageWarning{dataplot}{\string\dtlplothandlermark\space
     found outside \string\DTLplot}%
  \pgfplothandlermark{#1}%
}
```

TLplotatendtikz \DTLplotatendtikz is a hook to insert stuff at the end of the tikzpicture environment.

```
\newcommand*{\DTLplotatendtikz}{}
```

Plot settings. The database key for the *x* value is given by the x setting:

```
\define@key{dataplot}{x}{%
\def\dtl@xkey{#1}}
```

The database key for the *y* value is given by the y setting:

```
\define@key{dataplot}{y}{%
\def\dtl@ykey{#1}}
```

The list of plot mark colours is given by the markcolors setting. (This should be a comma separated list of colour names.)

```
\define@key{dataplot}{markcolors}{%
\def\DTLplotmarkcolors{#1}}
```

The list of plot line colours is given by the linecolors setting. (This should be a comma separated list of colour names.)

```
\define@key{dataplot}{linecolors}{%
\def\DTLplotlinecolors{#1}}
```

The list of plot mark and line colours is given by the colors setting. (This should be a comma separated list of colour names.)

```
\define@key{dataplot}{colors}{%
\def\DTLplotmarkcolors{#1}%
\def\DTLplotlinecolors{#1}}
```

The list of plot marks is given by the marks setting. (This should be a comma separated list of code that generates pgf plot marks.)

```
\define@key{dataplot}{marks}{%
\def\DTLplotmarks{#1}}
```

The list of plot line styles is given by the lines setting. (This should be a comma separated list of code that sets the line style.) An empty set will create solid lines.

```
\define@key{dataplot}{lines}{%
\def\DTLplotlines{#1}}
```

The total width of the plot is given by the width setting.

```
\define@key{dataplot}{width}{%
\setlength\DTLplotwidth{#1}}
```

The total height of the plot is given by the height setting.

```
\define@key{dataplot}{height}{%
\setlength\DTLplotheight{#1}}
```

Determine whether to show lines, markers or both

```
\define@choicekey{dataplot}{style}[\val\nr]{both,lines,markers}{%
\ifcase\nr\relax
 \DTLshowlinestrue
 \DTLshowmarkerstrue
\or
 \DTLshowlinestrue
 \DTLshowmarkersfalse
\or
 \DTLshowmarkerstrue
 \DTLshowlinesfalse
\fi}
```

Determine whether or not to display the axes

```
\define@choicekey{dataplot}{axes}[\val\nr]{both,x,y,none}[both]{%
\ifcase\nr\relax
 % both
 \DTLxaxistrue
 \DTLxticstrue
 \DTLyaxistrue
 \DTLyticstrue
\or % x
 \DTLxaxistrue
 \DTLxticstrue
 \DTLyaxisfalse
 \DTLyticsfalse
\or % y
 \DTLxaxisfalse
 \DTLxticsfalse
 \DTLyaxistrue
 \DTLyticstrue
\or % none
 \DTLxaxisfalse
 \DTLxticsfalse
 \DTLyaxisfalse
 \DTLyticsfalse
\fi
}
```

\ifDTLbox Enclose plot in a box

```
\define@boolkey{dataplot}[DTL]{box}[true]{}
\DTLboxfalse
```

\ifDTLxticstrue Condition to determine whether to show the *x* tick marks

```
\define@boolkey{dataplot}[DTL]{xtics}[true]{}
\DTLxticstrue
```

\ifDTLyticstrue Condition to determine whether to show the *y* tick marks

```
\define@boolkey{dataplot}[DTL]{ytics}[true]{}
\DTLyticstrue
```

ifDTLxminortics Condition to determine whether to show the *x* minor tick marks

```
\define@boolkey{dataplot}[DTL]{xminortics}[true]{%
\ifDTLxminortics \DTLxticstrue\fi}
\DTLxminorticsfalse
```

ifDTLyminortics Condition to determine whether to show the *y* minor tick marks

```
\define@boolkey{dataplot}[DTL]{yminortics}[true]{%
\ifDTLyminortics \DTLyticstrue\fi}
\DTLyminorticsfalse
```

\ifDTLgrid Determine whether to draw the grid

```
\define@boolkey{dataplot}[DTL]{grid}[true]{}
```

Determine whether the *x* tick marks should point in or out:

```
\define@choicekey{dataplot}{xticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLxticsintrue
\or
 \DTLxticsinfalse
\fi
}
```

Determine whether the *y* tick marks should point in or out:

```
\define@choicekey{dataplot}{yticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLyticsintrue
\or
 \DTLyticsinfalse
\fi
}
```

Determine whether the *x* and *y* tick marks should point in or out;

```
\define@choicekey{dataplot}{ticdir}[\val\nr]{in,out}{%
\ifcase\nr\relax
 \DTLxticsintrue
 \DTLyticsintrue
\or
 \DTLxticsinfalse
 \DTLyticsinfalse
\fi
}
```

Set the bounds of the graph (value must be in the form ⟨*min x*⟩,⟨*min y*⟩,⟨*max x*⟩,⟨*max y*⟩ (bounds overrides minx, miny, maxx and maxy settings.)

```
\define@key{dataplot}{bounds}{%
\def\dtl@bounds{#1}}
\let\dtl@bounds=\relax
```

Set only the lower *x* bound

```
\define@key{dataplot}{minx}{%
\def\dtl@minx{#1}}
\let\dtl@minx=\relax
```

Set only the upper *x* bound:

```
\define@key{dataplot}{maxx}{%
\def\dtl@maxx{#1}}
\let\dtl@maxx=\relax
```

Set only the lower *y* bound:

```
\define@key{dataplot}{miny}{%
\def\dtl@miny{#1}}
\let\dtl@miny=\relax
```

Set only the upper *y* bound:

```
\define@key{dataplot}{maxy}{%
\def\dtl@maxy{#1}}
\let\dtl@maxy=\relax
```

Define list of points for *x* ticks. (Must be a comma separated list of decimal numbers.)

```
\define@key{dataplot}{xticpoints}{%
\def\dtl@xticlist{#1}\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticlist=\relax
```

Define list of points for *y* ticks. (Must be a comma separated list of decimal numbers.)

```
\define@key{dataplot}{yticpoints}{%
\def\dtl@yticlist{#1}\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticlist=\relax
```

Define a the gap between *x* tick marks (xticpoints overrides xticgap)

```
\define@key{dataplot}{xticgap}{\def\dtl@xticgap{#1}%
\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticgap=\relax
```

Define a the gap between y tick marks (yticpoints overrides yticgap)

```
\define@key{dataplot}{yticgap}{\def\dtl@yticgap{#1}%
\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticgap=\relax
```

Define comma separated list of labels for x ticks.

```
\define@key{dataplot}{xticlabels}{%
\def\dtl@xticlabels{#1}\DTLxticstrue\DTLxaxistrue}
\let\dtl@xticlabels=\relax
```

Define comma separated list of labels for y ticks.

```
\define@key{dataplot}{yticlabels}{%
\def\dtl@yticlabels{#1}\DTLyticstrue\DTLyaxistrue}
\let\dtl@yticlabels=\relax
```

Define x axis label

```
\define@key{dataplot}{xlabel}{%
\def\dtl@xlabel{#1}}
\let\dtl@xlabel=\relax
```

Define y axis label

```
\define@key{dataplot}{ylabel}{%
\def\dtl@ylabel{#1}}
\let\dtl@ylabel=\relax
```

The legend setting may be one of: `none` (don't show it), `north`, `northeast`, `east`, `southeast`, `south`, `southwest`, `west`, or `northwest`. These set the count register `\dtl@legendsetting`.

```
\define@choicekey{dataplot}{legend}[\val\nr]{none,north,northeast,%
east,southeast,south,southwest,west,northwest}[northeast]{%
\dtl@legendsetting=\nr\relax
}
```

Legend labels (comma separated list). If omitted, the database name is used.

```
\define@key{dataplot}{legendlabels}{\def\dtl@legendlabels{#1}}
```

\DTLplot

```
\DTLplot[⟨condition⟩]{⟨db list⟩}{⟨settings⟩}
```

Creates a plot (inside a tikzpicture environment) of all the data given in the databases listed in ⟨*db list*⟩.

```
\newcommand*{\DTLplot}[3][\boolean{true}]{%
\bgroup
  \let\dtl@xkey=\relax
  \let\dtl@ykey=\relax
  \let\dtl@legendlabels=\relax
  \setkeys{dataplot}{#3}%
  \let\dtl@plotmarklist=\DTLplotmarks
  \let\dtl@plotlinelist=\DTLplotlines
  \let\dtl@plotmarkcolorlist=\DTLplotmarkcolors
```

```
    \let\dtl@plotlinecolorlist=\DTLplotlinecolors
    \def\dtl@legend{}%
    \ifx\dtl@legendlabels\relax
     \edef\dtl@legendlabels{#2}%
    \fi
    \ifx\dtl@xkey\relax
      \PackageError{dataplot}{Missing x setting for
        \string\DTLplot}{}%
    \else
      \ifx\dtl@ykey\relax
        \PackageError{dataplot}{Missing y setting for
        \string\DTLplot}{}%
      \else
```

If user didn't specified bounds, compute the maximum and minimum x and y values over all the databases listed.

```
        \ifx\dtl@bounds\relax
          \DTLcomputebounds[#1]{#2}{\dtl@xkey}{\dtl@ykey}
              {\DTLminX}{\DTLminY}{\DTLmaxX}{\DTLmaxY}%
          \ifx\dtl@minx\relax
          \else
             \let\DTLminX=\dtl@minx
          \fi
          \ifx\dtl@maxx\relax
          \else
             \let\DTLmaxX=\dtl@maxx
          \fi
          \ifx\dtl@miny\relax
          \else
             \let\DTLminY=\dtl@miny
          \fi
          \ifx\dtl@maxy\relax
          \else
             \let\DTLmaxY=\dtl@maxy
          \fi
```

Otherwise extract information from `\dtl@bounds`

```
        \else
          \expandafter\dtl@getbounds\dtl@bounds\@nil
        \fi
```

Determine scaling factors and offsets. The x-scale factor is given by:

$$s_x = \frac{W}{x_{\max} - x_{\min}}$$

where W is the plot width. The x offset is $-s_x x_{\min}$. Similarly for y.

```
        \@dtl@tmpcount=\DTLplotwidth
        \divide\@dtl@tmpcount by 65536\relax
        \dtlsub{\dtl@dx}{\DTLmaxX}{\DTLminX}%
        \dtldiv{\dtl@scale@x}{\number\@dtl@tmpcount}{\dtl@dx}%
```

```
 \dtlmul{\dtl@offset@x}{-\dtl@scale@x}{\DTLminX}%
 \@dtl@tmpcount=\DTLplotheight
 \divide\@dtl@tmpcount by 65536\relax
 \dtlsub{\dtl@dy}{\DTLmaxY}{\DTLminY}%
 \dtldiv{\dtl@scale@y}{\number\@dtl@tmpcount}{\dtl@dy}%
 \dtlmul{\dtl@offset@y}{-\dtl@scale@y}{\DTLminY}%
```

If x tics specified, construct a list of x tic points if not already specified.

```
\ifDTLxtics
  \ifx\dtl@xticlist\relax
    \ifx\dtl@xticgap\relax
```

Get the min tick gap in data co-ordinates

```
    \dtlsub{\dtl@mingap}{\number\DTLmintickgap}{\dtl@offset@x}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{\dtl@scale@x}%
    \dtldiv{\dtl@mingap}{\dtl@mingap}{65536}%
```

construct tick list

```
      \dtl@constructticklist\DTLminX\DTLmaxX
        \dtl@mingap\dtl@xticlist
    \else
      \DTLifFPopenbetween{0}{\DTLminX}{\DTLmaxX}{%
        \dtl@constructticklistwithgapz
          \DTLminX\DTLmaxX\dtl@xticlist\dtl@xticgap}{%
        \dtl@constructticklistwithgap
          \DTLminX\DTLmaxX\dtl@xticlist\dtl@xticgap}%
    \fi
  \fi
```

Construct a list of *x* minor tick points if required

```
  \let\dtl@xminorticlist\@empty
  \ifDTLxminortics
    \let\dtl@prevtick=\relax
    \@for\dtl@nexttick:=\dtl@xticlist\do{%
      \ifx\dtl@prevtick\relax
      \else
        \dtl@constructminorticklist
           \dtl@prevtick\dtl@nexttick\dtl@scale@x\dtl@xminorticlist
      \fi
      \let\dtl@prevtick=\dtl@nexttick
    }%
  \fi
```

Determine the height of the *x* tick labels.

```
  \ifx\dtl@xticlabels\relax
    \settoheight{\dtl@xticlabelheight}{\dtl@xticlist}%
  \else
    \settoheight{\dtl@xticlabelheight}{\dtl@xticlabels}%
   \fi
\else
  \setlength{\dtl@xticlabelheight}{0pt}%
\fi
```

If y tics specified, construct a list of y tic points if not already specified.

```
    \setlength{\dtl@yticlabelwidth}{0pt}%
    \ifDTLytics
      \ifx\dtl@yticlist\relax
        \ifx\dtl@yticgap\relax
```

Get the min tick gap in data co-ordinates

```
          \dtlsub{\dtl@mingap}{\number\DTLmintickgap}{\dtl@offset@y}%
          \dtldiv{\dtl@mingap}{\dtl@mingap}{\dtl@scale@y}%
          \dtldiv{\dtl@mingap}{\dtl@mingap}{65536}%
```

construct tick list

```
          \dtl@constructticklist\DTLminY\DTLmaxY
            \dtl@mingap\dtl@yticlist
          \else
            \DTLifFPopenbetween{0}{\DTLminY}{\DTLmaxY}{%
              \dtl@constructticklistwithgapz
                \DTLminY\DTLmaxY\dtl@yticlist\dtl@yticgap}{%
              \dtl@constructticklistwithgap
                \DTLminY\DTLmaxY\dtl@yticlist\dtl@yticgap}%
          \fi
        \fi
```

Construct a list of *y* minor tick points if required

```
        \let\dtl@yminorticlist\@empty
        \ifDTLyminortics
          \let\dtl@prevtick=\relax
          \@for\dtl@nexttick:=\dtl@yticlist\do{%
            \ifx\dtl@prevtick\relax
            \else
              \dtl@constructminorticklist
                 \dtl@prevtick\dtl@nexttick\dtl@scale@y\dtl@yminorticlist
            \fi
            \let\dtl@prevtick=\dtl@nexttick
          }%
        \fi
```

Determine the width of the *y* tick labels.

```
        \ifx\dtl@ylabel\relax
        \else
          \ifx\dtl@yticlabels\relax
            \@for\dtl@thislabel:=\dtl@yticlist\do{%
              \dtlround{\dtl@thislabel}{\dtl@thislabel}
                      {\c@DTLplotroundYvar}%
              \settowidth{\dtl@tmplength}{\dtl@thislabel}%
              \ifdim\dtl@tmplength>\dtl@yticlabelwidth
                \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
              \fi
            }%
          \else
            \@for\dtl@thislabel:=\dtl@yticlabels\do{%
              \settowidth{\dtl@tmplength}{\dtl@thislabel}%
              \ifdim\dtl@tmplength>\dtl@yticlabelwidth
                \setlength{\dtl@yticlabelwidth}{\dtl@tmplength}%
              \fi
            }%
          \fi
        \fi
      \fi
```

Start the picture.

```
\begin{tikzpicture}
```

Set the *x* and *y* unit vectors.

```
\pgfsetxvec{\pgfpoint{1pt}{0pt}}%
\pgfsetyvec{\pgfpoint{0pt}{1pt}}%
```

Set the transformation matrix, so user can plot things using the data co-ordinate space, but scope it, so it doesn't affect any plot marks later

```
\begin{scope}
\pgftransformcm{\dtl@scale@x}{0}{0}{\dtl@scale@y}%
  {\pgfpoint{\dtl@offset@x pt}{\dtl@offset@y pt}}%
```

Add any extra information the user requires

```
\let\dtlplothandlermark\@dtlplothandlermark
\DTLplotatbegintikz
```

Determine whether to put a box around the plot

```
\ifDTLbox
  \draw (\DTLminX,\DTLminY) -- (\DTLmaxX,\DTLminY) --
      (\DTLmaxX,\DTLmaxY) -- (\DTLminX,\DTLmaxY) --
        cycle;
\else
```

Plot *x* axis if required.

```
   \ifDTLxaxis
     \expandafter\draw\expandafter[\DTLXAxisStyle]
     (\DTLminX,\DTLminY) -- (\DTLmaxX,\DTLminY);
   \fi
```

Plot *y* axis if required.

```
   \ifDTLyaxis
     \expandafter\draw\expandafter[\DTLYAxisStyle]
     (\DTLminX,\DTLminY) -- (\DTLminX,\DTLmaxY);
   \fi
\fi
```

Plot grid if required

```
\ifDTLgrid
   \ifDTLxminortics
     \@for\dtl@thistick:=\dtl@xminorticlist\do{%
       \expandafter\draw\expandafter[\DTLminorgridstyle]
      (\dtl@thistick,\DTLminY) -- (\dtl@thistick,\DTLmaxY);
     }%
   \fi
   \ifDTLyminortics
     \@for\dtl@thistick:=\dtl@yminorticlist\do{%
       \expandafter\draw\expandafter[\DTLminorgridstyle]
      (\DTLminX,\dtl@thistick) -- (\DTLmaxX,\dtl@thistick);
     }%
   \fi
   \@for\dtl@thistick:=\dtl@xticlist\do{%
     \expandafter\draw\expandafter[\DTLmajorgridstyle]
      (\dtl@thistick,\DTLminY) -- (\dtl@thistick,\DTLmaxY);
   }%
   \@for\dtl@thistick:=\dtl@yticlist\do{%
     \expandafter\draw\expandafter[\DTLmajorgridstyle]
      (\DTLminX,\dtl@thistick) -- (\DTLmaxX,\dtl@thistick);
   }%
\fi
```

Plot *x* tics if required.

```
  \ifDTLxtics
```

Get tick length in terms of canvas co-ordinates

```
    \dtlsub{\dtl@ticklength}{\number\DTLticklength}{-\dtl@offset@y}%
    \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@y}%
    \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
```

Get tick label offset in terms of canvas co-ordinates

```
    \addtolength\dtl@xticlabelheight{\DTLticklabeloffset}%
    \dtlsub{\dtl@ticlabeloffset}{\number\dtl@xticlabelheight}{-\dtl@offset@y}%
    \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@y}%
    \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
```

Iterate through tick list.

```
    \@for\dtl@thistick:=\dtl@xticlist\do{%
```

Store tick label in \dtl@thislabel

```
      \let\dtl@thisticklabel\dtl@thistick
      \ifx\dtl@xticlabels\relax
        \dtlround{\dtl@thislabel}{\dtl@thistick}
                {\c@DTLplotroundXvar}%
      \else
        \dtl@chopfirst\dtl@xticlabels\dtl@thislabel\dtl@rest
        \let\dtl@xticlabels=\dtl@rest
      \fi
```

Draw tick.

```
      \ifDTLxticsin
        \draw (\dtl@thistick,\DTLminY) -- ++(0,\dtl@ticklength);
        \draw (\dtl@thistick,\DTLminY)
            ++ (0,-\dtl@ticlabeloffset) node {\dtl@thislabel};
      \else
        \draw (\dtl@thistick,\DTLminY) -- ++(0,-\dtl@ticklength)
```

```
              ++ (0,-\dtl@ticlabeloffset) node {\dtl@thislabel};
        \fi
```

Draw opposite tick, if box setting is on.

```
        \ifDTLbox
          \ifDTLxticsin
            \draw (\dtl@thistick,\DTLmaxY) -- ++(0,-\dtl@ticklength);
          \else
          \draw (\dtl@thistick,\DTLmaxY) -- ++(0,\dtl@ticklength);
          \fi
        \fi
      }%
     \fi
```

Plot *x* label if required.

```
  \ifx\dtl@xlabel\relax
  \else
```

Get baseline in terms of canvas co-ordinates

```
      \dtladd{\dtl@x}{\number\baselineskip}{\dtl@offset@y}%
      \dtldiv{\dtl@x}{\dtl@x}{\dtl@scale@y}%
      \dtldiv{\dtl@x}{\dtl@x}{65536}%
      \dtladd{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@x}%
```

Get halfway position

```
      \dtlmul{\dtl@x}{\dtl@dx}{0.5}%
      \draw (\DTLminX,\DTLminY) ++(\dtl@x,-\dtl@ticlabeloffset)
         node[anchor=north] {\dtl@xlabel};
  \fi
```

Plot the *x* minor ticks if required

```
  \ifDTLxminortics
```

Get tick length in terms of canvas co-ordinates

```
      \dtlsub{\dtl@ticklength}{\number\DTLminorticklength}{-\dtl@offset@y}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@y}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
```

Iterate through minor ticks.

```
      \@for\dtl@thistick:=\dtl@xminorticlist\do{%
        \ifDTLxticsin
          \draw (\dtl@thistick,\DTLminY) -- ++(0,\dtl@ticklength);
          \draw (\dtl@thistick,\DTLminY)
              ++ (0,-\dtl@ticlabeloffset) node[anchor=north] {\dtl@thislabel};
        \else
          \draw (\dtl@thistick,\DTLminY) -- ++(0,-\dtl@ticklength)
              ++ (0,-\dtl@ticlabeloffset) node[anchor=north] {\dtl@thislabel};
        \fi
```

Draw opposite tick, if box setting is on.

```
        \ifDTLbox
          \ifDTLxticsin
```

```
         \draw (\dtl@thistick,\DTLmaxY) -- ++(0,-\dtl@ticklength);
       \else
       \draw (\dtl@thistick,\DTLmaxY) -- ++(0,\dtl@ticklength);
       \fi
     \fi
  }%
\fi
```

Plot *y* tics if required.

```
\ifDTLytics
```

Get tick length in terms of canvas co-ordinates

```
   \dtlsub{\dtl@ticklength}{\number\DTLticklength}{-\dtl@offset@x}%
   \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@x}%
   \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
```

Get tick label offset in terms of canvas co-ordinates

```
   \dtladd{\dtl@ticlabeloffset}{\number\DTLticklabeloffset}{0}%
   \dtlsub{\dtl@ticlabeloffset}{\number\DTLticklabeloffset}{-\dtl@offset@x}%
   \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@x}%
   \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
```

Iterate through tick list.

```
   \@for\dtl@thistick:=\dtl@yticlist\do{%
```

Store tick label in `\dtl@thislabel`

```
     \let\dtl@thisticklabel\dtl@thistick
     \ifx\dtl@yticlabels\relax
       \dtlround{\dtl@thislabel}{\dtl@thistick}
               {\c@DTLplotroundXvar}%
     \else
       \dtl@chopfirst\dtl@yticlabels\dtl@thislabel\dtl@rest
       \let\dtl@yticlabels=\dtl@rest
     \fi
```

Draw tick.

```
     \ifDTLyticsin
       \draw (\DTLminX,\dtl@thistick) -- ++(\dtl@ticklength,0);
       \draw (\DTLminX,\dtl@thistick)
           ++ (-\dtl@ticlabeloffset,0) node[anchor=east] {\dtl@thislabel};
     \else
       \draw (\DTLminX,\dtl@thistick) -- ++(-\dtl@ticklength,0)
           ++ (-\dtl@ticlabeloffset,0) node[anchor=east] {\dtl@thislabel};
     \fi
```

Draw opposite tick, if box setting is on.

```
     \ifDTLbox
       \ifDTLyticsin
         \draw (\DTLmaxX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
       \else
       \draw (\DTLmaxX,\dtl@thistick) -- ++(\dtl@ticklength,0);
       \fi
```

```
        \fi
      }%
    \fi
```

Plot *y* label if required.

```
    \ifx\dtl@ylabel\relax
    \else
      \setlength{\dtl@tmplength}{\baselineskip}%
      \addtolength{\dtl@tmplength}{\dtl@yticlabelwidth}%
      \addtolength{\dtl@tmplength}{\DTLticklabeloffset}%
      \dtlsub{\dtl@ticlabeloffset}{\number\dtl@tmplength}{-\dtl@offset@x}%
      \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{\dtl@scale@x}%
      \dtldiv{\dtl@ticlabeloffset}{\dtl@ticlabeloffset}{65536}%
```

Get halfway position

```
      \dtlmul{\dtl@y}{\dtl@dy}{0.5}%
      \draw (\DTLminX,\DTLminY) ++(-\dtl@ticlabeloffset,\dtl@y)
         node[rotate=90,anchor=south] {\dtl@ylabel};
    \fi
```

Plot the *y* minor ticks if required

```
    \ifDTLyminortics
```

Get tick length in terms of canvas co-ordinates

```
      \dtlsub{\dtl@ticklength}{\number\DTLminorticklength}{-\dtl@offset@x}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{\dtl@scale@x}%
      \dtldiv{\dtl@ticklength}{\dtl@ticklength}{65536}%
```

Iterate through minor ticks.

```
      \@for\dtl@thistick:=\dtl@yminorticlist\do{%
        \ifDTLyticsin
          \draw (\DTLminX,\dtl@thistick) -- ++(\dtl@ticklength,0);
        \else
          \draw (\DTLminX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
        \fi
```

Draw opposite tick, if box setting is on.

```
        \ifDTLbox
          \ifDTLyticsin
            \draw (\DTLmaxX,\dtl@thistick) -- ++(-\dtl@ticklength,0);
          \else
          \draw (\DTLmaxX,\dtl@thistick) -- ++(\dtl@ticklength,0);
          \fi
        \fi
     }%
    \fi
```

End the transformation scope. (Don’t want marker shapes to be scaled or skewed.)

```
  \end{scope}
```

Iterate through each database

```
     \@for\dtl@thisdb:=#2\do{%
```

Get the current plot mark colour.

```
\ifx\dtl@plotmarkcolorlist\@empty
  \let\dtl@plotmarkcolorlist=\DTLplotmarkcolors
\fi
\dtl@chopfirst\dtl@plotmarkcolorlist\dtl@thisplotmarkcolor
   \dtl@remainder
\let\dtl@plotmarkcolorlist=\dtl@remainder
```

Get the current plot mark, and store in `\dtl@mark`

```
\ifDTLshowmarkers
  \ifx\dtl@plotmarklist\@empty
     \let\dtl@plotmarklist=\DTLplotmarks
  \fi
  \dtl@chopfirst\dtl@plotmarklist\dtl@thisplotmark
     \dtl@remainder
  \let\dtl@plotmarklist=\dtl@remainder
  \ifx\dtl@thisplotmark\relax
    \let\dtl@mark=\relax
  \else
    \expandafter\toks@\expandafter{\dtl@thisplotmark}%
    \ifx\dtl@thisplotmarkcolor\@empty
      \edef\dtl@mark{\the\toks@}%
    \else
      \edef\dtl@mark{%
          \noexpand\color{\dtl@thisplotmarkcolor}%
        \the\toks@}%
    \fi
  \fi
\else
  \let\dtl@mark=\relax
\fi
```

Get the current plot line colour.

```
\ifx\dtl@plotlinecolorlist\@empty
  \let\dtl@plotlinecolorlist=\DTLplotlinecolors
\fi
\dtl@chopfirst\dtl@plotlinecolorlist\dtl@thisplotlinecolor
   \dtl@remainder
\let\dtl@plotlinecolorlist=\dtl@remainder
```

Get the current line style, and store in `\dtl@linestyle`

```
\ifDTLshowlines
  \ifx\dtl@plotlinelist\@empty
     \let\dtl@plotlinelist=\DTLplotlines
  \fi
  \dtl@chopfirst\dtl@plotlinelist\dtl@thisplotline
     \dtl@remainder
  \let\dtl@plotlinelist=\dtl@remainder
  \expandafter\ifx\dtl@thisplotline\relax
  \let\dtl@linestyle=\relax
  \else
    \expandafter\toks@\expandafter{\dtl@thisplotline}%
    \ifx\dtl@thisplotlinecolor\@empty
      \edef\dtl@linestyle{\the\toks@}%
    \else
      \edef\dtl@linestyle{%
          \noexpand\color{\dtl@thisplotlinecolor}%
        \the\toks@}%
    \fi
  \fi
\else
  \let\dtl@linestyle=\relax
\fi
```

Append this plot setting to the legend.

```
\ifnum\dtl@legendsetting>0\relax
  \dtl@chopfirst\dtl@legendlabels\dtl@thislabel\dtl@rest
  \let\dtl@legendlabels=\dtl@rest
  \expandafter\toks@\expandafter{\dtl@mark}%
  \expandafter\@dtl@toks\expandafter{\dtl@linestyle}%
  \edef\dtl@addtolegend{\noexpand\DTLaddtoplotlegend
    {\the\toks@}{\the\@dtl@toks}{\dtl@thislabel}}%
  \dtl@addtolegend
\fi
```

Store stream in `\dtl@stream`

```
\def\dtl@stream{\pgfplotstreamstart}%
```

Only plot points that lie inside bounds.

```
\@sDTLforeach[#1]{\dtl@thisdb}{\dtl@x=\dtl@xkey,%
    \dtl@y=\dtl@ykey}{%
  \DTLconverttodecimal{\dtl@x}{\dtl@decx}%
  \DTLconverttodecimal{\dtl@y}{\dtl@decy}%
  \ifthenelse{%
    \DTLisclosedbetween{\dtl@x}{\DTLminX}{\DTLmaxX}%
    \and
    \DTLisclosedbetween{\dtl@y}{\DTLminY}{\DTLmaxY}%
    }%
    {%
      \expandafter\toks@\expandafter{\dtl@stream}%
```

Apply transformation to co-ordinates

```
      \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
      \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
      \dtlround{\dtl@decx}{\dtl@decx}{1}%
      \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
      \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
      \dtlround{\dtl@decy}{\dtl@decy}{1}%
      \edef\dtl@stream{\the\toks@
        \noexpand\pgfplotstreampoint
         {\noexpand\pgfpointxy{\dtl@decx}{\dtl@decy}}}%
```

```
    }{}%
  }%
  \expandafter\toks@\expandafter{\dtl@stream}%
  \edef\dtl@stream{\the\toks@\noexpand\pgfplotstreamend}%
```

End plot stream and draw path.

```
  \ifx\dtl@linestyle\relax
  \else
    \begin{scope}
    \dtl@linestyle
    \pgfplothandlerlineto
    \dtl@stream
    \pgfusepath{stroke}
    \end{scope}
  \fi
  \ifx\dtl@mark\relax
  \else
    \begin{scope}
    \pgfplothandlermark{\dtl@mark}%
    \dtl@stream
    \pgfusepath{stroke}
    \end{scope}
  \fi
  }%
```

Plot legend if required.

```
  \ifcase\dtl@legendsetting
% none
  \or % north
   \dtlmul{\dtl@decx}{\dtl@dx}{0.5}%
   \dtladd{\dtl@decx}{\DTLminX}{\dtl@decx}%
   \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
   \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
   \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
   \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
   \draw (\dtl@decx,\dtl@decy) ++(0,-\DTLlegendyoffset)
      node[anchor=north]
      {\DTLformatlegend
        {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
      };
  \or % north east
   \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
   \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
   \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
   \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
   \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,-\DTLlegendyoffset)
      node[anchor=north east]
      {\DTLformatlegend
        {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
      };
```

```
\or % east
 \dtlmul{\dtl@decy}{\dtl@dy}{0.5}%
 \dtladd{\dtl@decy}{\DTLminY}{\dtl@decy}%
 \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,0)
    node[anchor=east]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south east
 \dtlmul{\dtl@decx}{\DTLmaxX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(-\DTLlegendxoffset,\DTLlegendyoffset)
    node[anchor=south east]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south
 \dtlmul{\dtl@decx}{\dtl@dx}{0.5}%
 \dtladd{\dtl@decx}{\DTLminX}{\dtl@decx}%
 \dtlmul{\dtl@decx}{\dtl@decx}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(0,\DTLlegendyoffset)
    node[anchor=south]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % south west
 \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
 \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
 \dtlmul{\dtl@decy}{\DTLminY}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
 \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,\DTLlegendyoffset)
    node[anchor=south west]
    {\DTLformatlegend
      {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
    };
\or % west
 \dtlmul{\dtl@decy}{\dtl@dy}{0.5}%
 \dtladd{\dtl@decy}{\DTLminY}{\dtl@decy}%
 \dtlmul{\dtl@decy}{\dtl@decy}{\dtl@scale@y}%
 \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
```

```
         \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
         \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
         \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,0)
            node[anchor=west]
            {\DTLformatlegend
              {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
            };
        \or % north west
         \dtlmul{\dtl@decx}{\DTLminX}{\dtl@scale@x}%
         \dtladd{\dtl@decx}{\dtl@decx}{\dtl@offset@x}%
         \dtlmul{\dtl@decy}{\DTLmaxY}{\dtl@scale@y}%
         \dtladd{\dtl@decy}{\dtl@decy}{\dtl@offset@y}%
         \draw (\dtl@decx,\dtl@decy) ++(\DTLlegendxoffset,-\DTLlegendyoffset)
            node[anchor=north west]
            {\DTLformatlegend
              {\begin{tabular}{cl}\dtl@legend\end{tabular}}%
            };
        \fi
```

Set the transformation matrix, so user can plot things using the data co-ordinate space

```
       \pgftransformcm{\dtl@scale@x}{0}{0}{\dtl@scale@y}%
         {\pgfpoint{\dtl@offset@x pt}{\dtl@offset@y pt}}%
```

End hook

```
       \let\dtlplothandlermark\@dtlplothandlermark
        \DTLplotatendtikz
       \end{tikzpicture}
    \fi
  \fi
\egroup
}
```

\dtl@getbounds Extract bounds:

```
\def\dtl@getbounds#1,#2,#3,#4\@nil{%
\def\DTLminX{#1}%
\def\DTLminY{#2}%
\def\DTLmaxX{#3}%
\def\DTLmaxY{#4}%
\dtlifnumgt{\DTLminX}{\DTLmaxX}
{%
 \PackageError{dataplot}{Min X > Max X in bounds #1,#2,#3,#4}{%
 The bounds must be specified as minX,minY,maxX,maxY}%
}{}%
\dtlifnumgt{\DTLminY}{\DTLmaxY}
{%
 \PackageError{dataplot}{Min Y > Max Y in bounds #1,#2,#3,#4}{%
 The bounds must be specified as minX,minY,maxX,maxY}%
}{}%
}
```

onstructticklist `\dtl@constructticklist{⟨min⟩}{⟨max⟩}{⟨min gap⟩}{⟨list⟩}`

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence.)

```
  \newcommand*{\dtl@constructticklist}[4]{%
    \DTLifFPopenbetween{0}{#1}{#2}%
    {%
```

Tick list straddles the origin.

```
      \dtlsub{\@dtl@width}{0}{#1}%
      \dtldiv{\@dtl@neggap}{\@dtl@width}{10}%
      \dtlifnumlt{\@dtl@neggap}{#3}%
      {%
        \edef\@dtl@neggap{#3}%
      }%
      {}%
      \dtldiv{\@dtl@posgap}{#2}{10}%
      \dtlifnumlt{\@dtl@posgap}{#3}%
      {%
        \edef\@dtl@posgap{#3}%
      }%
      {}%
      \dtlmax{\@dtl@gap}{\@dtl@neggap}{\@dtl@posgap}%
```

Don't construct a list if minimum gap is greater than plot width

```
      \dtlifnumgt{\@dtl@gap}{\@dtl@width}%
      {}%
      {%
        \dtl@constructticklistwithgapz{#1}{#2}{#4}{\@dtl@gap}%
      }%
    }%
    {%
```

Tick list doesn't straddle the origin.

```
      \dtlsub{\@dtl@width}{#2}{#1}%
      \dtldiv{\@dtl@gap}{\@dtl@width}{10}%
      \dtlifnumlt{\@dtl@gap}{#3}%
      {%
```

Don't construct a list if minimum gap is greater than plot width

```
        \dtlifnumgt{#3}{\@dtl@width}%
        {%
           \def#4{#1,#2}%
        }%
        {%
          \dtl@constructticklistwithgap{#1}{#2}{#4}{#3}%
        }
      }%
      {%
        \dtl@constructticklistwithgap{#1}{#2}{#4}{\@dtl@gap}%
      }%
    }%
  }
```

tticklistwithgap

```
\dtl@constructticklistwithgap{⟨min⟩}{⟨max⟩}{⟨list⟩}{⟨gap⟩}
```

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates.

```
\newcommand*{\dtl@constructticklistwithgap}[4]{%
\edef\@dtl@thistick{#1}%
\edef#3{#1}%
\dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
\whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{#2}}{%
  \expandafter\toks@\expandafter{\@dtl@thistick}%
  \edef#3{#3,\the\toks@}%
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
}%
\expandafter\toks@\expandafter{#2}%
\edef#3{#3,\the\toks@}%
}
```

ticklistwithgapz

```
\dtl@constructticklistwithgapz{⟨min⟩}{⟨max⟩}{⟨list⟩}{⟨gap⟩}
```

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the tick list straddles zero.

```
\newcommand*{\dtl@constructticklistwithgapz}[4]{%
  \edef\@dtl@thistick{0}%
  \edef#3{0}%
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
  \whiledo{\DTLisFPopenbetween{\@dtl@thistick}{0}{#2}}%
  {%
    \expandafter\toks@\expandafter{\@dtl@thistick}%
    \edef#3{#3,\the\toks@}%
    \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
  }%
  \expandafter\toks@\expandafter{#2}%
  \edef#3{#3,\the\toks@}%
  \dtlifnumeq{#1}{0}%
  {}%
  {%
    \edef\@dtl@thistick{0}%
    \dtlsub{\@dtl@thistick}{\@dtl@thistick}{#4}%
    \whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{0}}%
    {%
      \expandafter\toks@\expandafter{\@dtl@thistick}%
      \edef#3{\the\toks@,#3}%
      \dtlsub{\@dtl@thistick}{\@dtl@thistick}{#4}%
    }%
    \expandafter\toks@\expandafter{#1}%
    \edef#3{\the\toks@,#3}%
  }%
}
```

uctminorticklist

\dtl@constructminorticklist{⟨*min*⟩}{⟨*max*⟩}{⟨*scale factor*⟩}{⟨*list*⟩}

Constructs a list of minor tick points between ⟨*min*⟩ and ⟨*max*⟩ and append to ⟨*list*⟩ (a control sequence.)

```
\newcommand*{\dtl@constructminorticklist}[4]{%
  \dtlsub{\@dtl@width}{#2}{#1}%
  \dtlmul{\@dtl@width}{\@dtl@width}{#3}%
  \dtldiv{\@dtl@gap}{\@dtl@width}{10}%
  \setlength\dtl@tmplength{\@dtl@gap sp}%
  \ifdim\dtl@tmplength<\DTLminminortickgap
    \dtldiv{\@dtl@gap}{\@dtl@width}{4}%
    \setlength\dtl@tmplength{\@dtl@gap sp}%
    \ifdim\dtl@tmplength<\DTLminminortickgap
      \dtldiv{\@dtl@gap}{\@dtl@width}{2}%
      \setlength\dtl@tmplength{\@dtl@gap sp}%
      \ifdim\dtl@tmplength<\DTLminminortickgap
        \let\@dtl@gap=\@dtl@width
      \fi
    \fi
  \fi
  \dtldiv{\@dtl@gap}{\@dtl@gap}{#3}%
  \dtl@constructticklistwithgapex{#1}{#2}{\dtl@tmp}{\@dtl@gap}%
  \ifx#4\@empty
    \let#4=\dtl@tmp
  \else
    \expandafter\toks@\expandafter{#4}%
    \edef#4{#4,\dtl@tmp}%
  \fi
}
```

icklistwithgapex

\dtl@constructticklistwithgapex{⟨*min*⟩}{⟨*max*⟩}{⟨*list*⟩}{⟨*gap*⟩}

Constructs a list of tick points between ⟨*min*⟩ and ⟨*max*⟩ and store in ⟨*list*⟩ (a control sequence) using the gap given by ⟨*gap*⟩ where the gap is given in user co-ordinates. The end points are excluded from the list.

```
\newcommand*{\dtl@constructticklistwithgapex}[4]{%
\edef\@dtl@thistick{#1}%
\let#3=\@empty
\dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
\whiledo{\DTLisFPopenbetween{\@dtl@thistick}{#1}{#2}}{%
  \expandafter\toks@\expandafter{\@dtl@thistick}%
  \ifx#3\@empty
    \edef#3{\the\toks@}%
  \else
    \edef#3{#3,\the\toks@}%
  \fi
  \dtladd{\@dtl@thistick}{\@dtl@thistick}{#4}%
}%
}
```

Laddtoplotlegend

\DTLaddtoplotlegend{⟨*marker*⟩}{⟨*line style*⟩}{⟨*label*⟩}

Adds entry to legend.

```
\newcommand*{\DTLaddtoplotlegend}[3]{%
\def\dtl@legendline{}%
\ifx\relax#2\relax
\else
  \toks@{#2%
  \pgfpathmoveto{\pgfpoint{-10pt}{0pt}}%
  \pgfpathlineto{\pgfpoint{10pt}{0pt}}%
  \pgfusepath{stroke}}%
  \edef\dtl@legendline{\the\toks@}%
\fi
\ifx\relax#1\relax
\else
  \toks@{#1}%
  \expandafter\@dtl@toks\expandafter{\dtl@legendline}%
  \edef\dtl@legendline{\the\@dtl@toks\the\toks@}%
\fi
\expandafter\toks@\expandafter{\dtl@legendline}%
\ifx\dtl@legend\@empty
  \xdef\dtl@legend{\noexpand\tikz\the\toks@; \noexpand& #3}%
\else
  \expandafter\@dtl@toks\expandafter{\dtl@legend}%
  \xdef\dtl@legend{\the\@dtl@toks\noexpand\\%
    \noexpand\tikz\the\toks@; \noexpand& #3}%
\fi
}
```

10 person.sty

10.1 Package Declaration

Package identification:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{person}[2019/09/27 v2.32 (NLCT)]
```

Requires the ifthen package.

```
\RequirePackage{ifthen}
\RequirePackage{datatool}
```

10.2 Defining People

people Keep count of the number of people who have been defined:

```
\newcounter{people}
```

person Temporary counter

```
\newcounter{person}
```

\@people@list Keep a list of labels for each person who has been defined:

```
\newcommand*{\@people@list}{,}
```

get@firstperson Get the first person's name in `\@people@list`, and store in the argument (which must be a control sequence.)

```
\newcommand*{\@get@firstperson}[1]{%
  \expandafter\@@get@firstperson\@people@list,\@nil{#1}}
  \def\@@get@firstperson,#1,#2\@nil#3{%
  \def#3{#1}%
}
```

\malelabels List of labels that can be used to indicate that a person is male (when defining a person using `\newperson`).

```
\newcommand*{\malelabels}{male,Male,MALE,M,m}
```

\addmalelabel Adds a label to the list of male labels.

```
\newcommand*{\addmalelabel}[1]{%
  \expandafter\@dtl@toksA\expandafter{\malelabels}%
  \expandafter\@dtl@toksB\expandafter{#1}%
  \edef\malelabels{\the\@dtl@toksA,\the\@dtl@toksB}%
}
```

\addfemalelabel Adds a label to the list of female labels.

```
\newcommand*{\addfemalelabel}[1]{%
  \expandafter\@dtl@toksA\expandafter{\femalelabels}%
  \expandafter\@dtl@toksB\expandafter{#1}%
  \edef\femalelabels{\the\@dtl@toksA,\the\@dtl@toksB}%
}
```

\femalelabels List of labels that can be used to indicate that a person is female (when defining a person using \newperson).

```
\newcommand*{\femalelabels}{female,Female,FEMALE,F,f}
```

\ifmalelabel Determines if first argument is contained in the list of male labels. (One level expansion is performed on the first object in first argument.) If true does second argument, otherwise does third argument.

```
\newcommand{\ifmalelabel}[3]{%
   \expandafter\DTLifinlist\expandafter{#1}{\malelabels}{#2}{#3}%
}
```

\iffemalelabel Determines if first argument is contained in the list of female labels. (One level expansion is performed on the first object in first argument.) If true does second argument, otherwise does third argument.

```
\newcommand{\iffemalelabel}[3]{%
   \expandafter\DTLifinlist\expandafter{#1}{\femalelabels}{#2}{#3}%
}
```

\newperson Define a new person. The optional argument specifies a label with which to refer to that person. If omitted, anon is used. If more than one person is defined, the optional argument will be required to specify a unique label. The compulsory arguments are the person's full name, their familiar name and their gender.

```
\newcommand*{\newperson}[4][anon]{%
  \@ifundefined{person@#1@name}%
  {%
    \ifmalelabel{#4}%
    {%
      \expandafter\gdef\csname person@#1@gender\endcsname{male}%
    }%
    {%
      \iffemalelabel{#4}%
      {%
        \expandafter\gdef\csname person@#1@gender\endcsname{female}%
      }%
      {%
         \PackageError{person}{Unknown gender ‘#4’ for person
         ‘#1’}{Allowed gender labels are: \malelabels\space or
         \femalelabels}%
         \@namedef{person@#1@gender}{other}%
      }%
    }%
    \expandafter
      \protected@xdef\csname person@#1@fullname\endcsname{#2}%
    \expandafter
      \protected@xdef\csname person@#1@name\endcsname{#3}%
    \protected@xdef\@people@list{\@people@list#1,}%
    \stepcounter{people}%
  }%
  {%
    \PackageError{person}{Person ‘#1’ has already been defined}{}%
  }%
}
```

10.3 Remove People

\removeperson Removes person identified by their label from the list.

```
  \newcommand*{\removeperson}[1][anon]{%
    \edef\@person@label{#1}%
    \expandafter\@removeperson\expandafter{\@person@label}%
  }
```

The label has to be full expanded for the internal command.

```
  \newcommand*{\@removeperson}[1]{%
     \ifpersonexists{#1}%
     {%
```

Remove label from list of people.

```
       \def\@remove@person##1,#1,##2\@nil{%
         \def\@prsn@pre{##1}\def\@prsn@post{##2}}%
       \expandafter\@remove@person\@people@list\@nil
       \xdef\@people@list{\@prsn@pre,\@prsn@post}%
```

Decrement number of people:

```
       \addtocounter{people}{-1}%
```

Undefine associated control sequences:

```
       \expandafter\global\expandafter
         \let\csname person@#1@name\endcsname\undefined
       \expandafter\global\expandafter
         \let\csname person@#1@fullname\endcsname\undefined
       \expandafter\global\expandafter
         \let\csname person@#1@gender\endcsname\undefined
     }%
     {%
       \PackageError{person}{Can’t remove person ‘#1’: no such
       person}{}%
     }%
  }
```

\removepeople Removes the people listed.

```
\newcommand*{\removepeople}[1]{%
  \@for\@thisperson:=#1\do{%
    \ifx\@thisperson\@empty
    \else
      \expandafter\removeperson\expandafter[\@thisperson]%
    \fi
  }%
}
```

removeallpeople

Removes everyone.

```
\newcommand*{\removeallpeople}{%
  \@for\@thisperson:=\@people@list\do{%
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @name\endcsname\undefined
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @fullname\endcsname\undefined
    \expandafter\global\expandafter
      \let\csname person@\@thisperson @gender\endcsname\undefined
  }%
  \setcounter{people}{0}%
  \gdef\@people@list{,}%
}
```

10.4 Conditionals and Loops

\ifpersonexists

If person whose label is given by the first argument exists, then do the second argument otherwise do third argument.

```
\newcommand{\ifpersonexists}[3]{%
  \@ifundefined{person@#1@name}{#3}{#2}%
}
```

\ifmale

If the person given by the label in the first argument is male, do the second argument, otherwise do the third argument.

```
\newcommand{\ifmale}[3]{%
   \ifpersonexists{#1}%
   {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \ifx\@gender\@male@label
        #2%
      \else
        #3%
      \fi
   }%
   {%
      \PackageError{person}{Person ‘#1’ doesn’t exist.}{}%
   }%
}
\def\@male@label{male}
```

\ifallmale If all people listed in first argument are male, do the second argument otherwise do the third argument. If the first argument is omitted, all defined people are checked.

```
\newcommand{\ifallmale}[3][\@people@list]{%
 \@for\@thisperson:=#1\do{%
  \ifpersonexists{\@thisperson}%
  {%
   \edef\@gender{\csname person@\@thisperson @gender\endcsname}%
   \ifx\@gender\@male@label
   \else
    \@endfortrue
   \fi
  }%
  {%
   \PackageError{person}{Person ‘#1’ doesn’t exist.}{}%
  }%
 }%
 \if@endfor
  #3%
 \else
  #2%
 \fi
}
```

\iffemale If the person given by the label in the first argument is female, do the second argument, otherwise do the third argument.

```
\newcommand{\iffemale}[3]{%
  \ifpersonexists{#1}%
  {%
   \edef\@gender{\csname person@#1@gender\endcsname}%
   \ifx\@gender\@female@label
    #2%
   \else
    #3%
   \fi
  }%
  {%
   \PackageError{person}{Person ‘#1’ doesn’t exist.}{}%
  }%
}
\def\@female@label{female}
```

\ifallfemale If all people listed in first argument are female, do the second argument otherwise do the third argument.

```
\newcommand{\ifallfemale}[3][\@people@list]{%
 \@for\@thisperson:=#1\do{%
  \edef\@gender{\csname person@\@thisperson @gender\endcsname}%
  \ifx\@gender\@female@label
  \else
   \@endfortrue
```

```
    \fi
  }%
  \if@endfor
    #3%
  \else
    #2%
  \fi
}
```

\foreachperson

\foreachperson(⟨*name cs*⟩,⟨*full name cs*⟩,⟨*gender cs*⟩,⟨*label cs*⟩)\in {⟨*list*⟩}\do{⟨*body*⟩}

Iterates through list of people the \in{⟨*list*⟩} is optional. If omitted, the list of all defined people is used.

```
\def\foreachperson(#1,#2,#3,#4)#5{%
  \ifx#5\in
    \def\@do@foreachperson{\@foreachperson(#1,#2,#3,#4)#5}%
  \else
    \def\@do@foreachperson{%
      \@foreachperson(#1,#2,#3,#4)\in\@people@list#5}%
  \fi
  \@do@foreachperson
}
\long\def\@foreachperson(#1,#2,#3,#4)\in#5\do#6{%
  \@for#4:=#5\do{%
    \ifx#4\@empty
    \else
       \ifpersonexists{#4}%
       {%
         \expandafter
           \let\expandafter#1\csname person@#4@name\endcsname
         \expandafter
           \let\expandafter#2\csname person@#4@fullname\endcsname
         \expandafter
           \let\expandafter#3\csname person@#4@gender\endcsname
         \ifx#3\@male@label
           \let#3\malename
         \else
           \ifx#3\@female@label
             \let#3\femalename
           \fi
         \fi
         #6%
       }%
       {%
         \PackageError{person}{Person ‘#4’ doesn’t exist}{}%
       }%
      \fi
    }%
  }
```

10.5 Predefined Words

These commands should be redefined if you are writing in another language, but note that these are structured according to English grammar.

\malepronoun

```
\newcommand*{\malepronoun}{he}
```

\femalepronoun

```
\newcommand*{\femalepronoun}{she}
```

\pluralpronoun

```
\newcommand*{\pluralpronoun}{they}
```

\maleobjpronoun

```
\newcommand*{\maleobjpronoun}{him}
```

emaleobjpronoun

```
\newcommand*{\femaleobjpronoun}{her}
```

luralobjpronoun

```
\newcommand*{\pluralobjpronoun}{them}
```

\malepossadj

```
\newcommand*{\malepossadj}{his}
```

\femalepossadj

```
\newcommand*{\femalepossadj}{her}
```

\pluralpossadj

```
\newcommand*{\pluralpossadj}{their}
```

maleposspronoun

```
\newcommand*{\maleposspronoun}{his}
```

maleposspronoun

```
\newcommand*{\femaleposspronoun}{hers}
```

uralposspronoun

```
\newcommand*{\pluralposspronoun}{theirs}
```

\malechild

```
\newcommand*{\malechild}{son}
```

\femalechild

```
\newcommand*{\femalechild}{daughter}
```

\pluralchild

```
\newcommand*{\pluralchild}{children}
```

\malechildren

```
\newcommand*{\malechildren}{sons}
```

\femalechildren

```
\newcommand*{\femalechildren}{daughters}
```

\maleparent

```
\newcommand*{\maleparent}{father}
```

\femaleparent

```
\newcommand*{\femaleparent}{mother}
```

\pluralparent

```
\newcommand*{\pluralparent}{parents}
```

\malesibling

```
\newcommand*{\malesibling}{brother}
```

\femalesibling

```
\newcommand*{\femalesibling}{sister}
```

\pluralsibling

```
\newcommand*{\pluralsibling}{siblings}
```

\malesiblings

```
\newcommand*{\malesiblings}{brothers}
```

\femalesiblings

```
\newcommand*{\femalesiblings}{sisters}
```

\andname Define `\andname` if it hasn't already been defined:

```
\providecommand*{\andname}{and}
```

\malename

```
\newcommand*{\malename}{male}
```

\femalename

```
\newcommand*{\femalename}{female}
```

\personsep Separator to use between people (but not the between the last two).

```
\newcommand*{\personsep}{, }
```

`\personlastsep` Separator to use between last two people.

```
\newcommand*{\personlastsep}{\space\andname\space}
```

`\twopeoplesep` Separator to use when list only contains two people.

```
\newcommand*{\twopeoplesep}{\space\andname\space}
```

10.6 Displaying Information

`\personfullname` The person's full name can be displayed using `\personfullname[`⟨*label*⟩`]`, where ⟨*label*⟩ is the unique label used when defining that person. If ⟨*label*⟩ is omitted, anon is used.

```
\newcommand*{\personfullname}[1][anon]{%
  \@ifundefined{person@#1@fullname}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \csname person@#1@fullname\endcsname
  }%
}
```

`\peoplefullname` List all defined people's full names. This iterates through all labels in `\@people@list`.

```
\newcommand*{\peoplefullname}{%
  \setcounter{person}{1}%
  \@for\@thisperson:=\@people@list\do{%
    \ifthenelse{\equal{\@thisperson}{}}%
    {}%
    {%
      \personfullname[\@thisperson]%
      \stepcounter{person}%
      \ifnum\c@people=1\relax
      \else
        \ifnum\c@person=\c@people
          \ifnum\c@people=2\relax
            \twopeoplesep
          \else
            \personlastsep
          \fi
        \else
           \ifnum\c@person<\c@people
             \personsep
           \fi
        \fi
      \fi
    }%
  }%
}
```

\personname As \personfullname, but for the person’s familiar name.

```
\newcommand*{\personname}[1][anon]{%
  \@ifundefined{person@#1@name}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \csname person@#1@name\endcsname
  }%
}
```

\peoplename List all defined people’s familiar names. This iterates through all labels in \@people@list.

```
\newcommand*{\peoplename}{%
  \setcounter{person}{1}%
  \@for\@thisperson:=\@people@list\do{%
    \ifthenelse{\equal{\@thisperson}{}}%
    {}%
    {%
      \personname[\@thisperson]%
      \stepcounter{person}%
      \ifnum\c@people=1\relax
      \else
        \ifnum\c@person=\c@people
          \ifnum\c@people=2\relax
            \twopeoplesep
          \else
            \personlastsep
          \fi
        \else
           \ifnum\c@person<\c@people
             \personsep
           \fi
        \fi
      \fi
    }%
  }%
}
```

\personpronoun Display the pronoun (he/she) according to the person’s gender.

```
\newcommand*{\personpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender pronoun\endcsname
  }%
}
```

\Personpronoun As above, but make the first letter uppercase.

```
\newcommand*{\Personpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender pronoun\endcsname
  }%
}
```

\peoplepronoun If there is more than one person, \peoplepronoun will use \pluralpronoun, otherwise it will use \personpronoun.

```
\newcommand*{\peoplepronoun}{%
  \ifnum\c@people>1\relax
     \pluralpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \personpronoun[\@thisperson]%
  \fi
}
```

\Peoplepronoun As above, but first letter in upper case

```
\newcommand*{\Peoplepronoun}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \Personpronoun[\@thisperson]%
  \fi
}
```

ersonobjpronoun Display the objective pronoun (him/her) according to the person's gender.

```
\newcommand*{\personobjpronoun}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender objpronoun\endcsname
  }%
}
```

ersonobjpronoun As above, but make the first letter uppercase.

```
\newcommand*{\Personobjpronoun}[1][anon]{%
```

```
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender objpronoun\endcsname
  }%
}
```

eopleobjpronoun If there is more than one person, \peopleobjpronoun will use \pluralobjpronoun, otherwise it will use \personobjpronoun.

```
\newcommand*{\peopleobjpronoun}{%
  \ifnum\c@people>1\relax
     \pluralobjpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \personobjpronoun[\@thisperson]%
  \fi
}
```

eopleobjpronoun As above, but first letter in upper case

```
\newcommand*{\Peopleobjpronoun}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralobjpronoun
  \else
     \@get@firstperson{\@thisperson}%
     \Personobjpronoun[\@thisperson]%
  \fi
}
```

\personpssadj Display the possessive adjective (his/her) according to the person’s gender.

```
\newcommand*{\personpossadj}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender possadj\endcsname
  }%
}
```

\Personpossadj As above, but make the first letter uppercase.

```
\newcommand*{\Personpossadj}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter
    \MakeUppercase\csname\@gender possadj\endcsname
  }%
}
```

\peoplepossadj If there is more than one person, \peoplepossadj will use \pluralpossadj, otherwise it will use \personpossadj.

```
  \newcommand*{\peoplepossadj}{%
    \ifnum\c@people>1\relax
       \pluralpossadj
    \else
       \@get@firstperson{\@thisperson}%
       \personpossadj[\@thisperson]%
    \fi
  }
```

\Peoplepossadj As above, but first letter in upper case

```
  \newcommand*{\Peoplepossadj}{%
    \ifnum\c@people>1\relax
       \expandafter\MakeUppercase\pluralpossadj
    \else
       \@get@firstperson{\@thisperson}%
       \Personpossadj[\@thisperson]%
    \fi
  }
```

rsonposspronoun Display possessive pronoun (his/hers) according to the person's gender.

```
  \newcommand*{\personposspronoun}[1][anon]{%
    \@ifundefined{person@#1@gender}%
    {%
      \PackageError{person}{Person '#1' has not been defined}{}%
    }%
    {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \csname\@gender posspronoun\endcsname
    }%
  }
```

rsonposspronoun As above, but make the first letter uppercase.

```
  \newcommand*{\Personposspronoun}[1][anon]{%
    \@ifundefined{person@#1@gender}%
    {%
      \PackageError{person}{Person '#1' has not been defined}{}%
    }%
    {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \expandafter\expandafter\expandafter
      \MakeUppercase\csname\@gender posspronoun\endcsname
    }%
  }
```

opleposspronoun If there is more than one person, \peopleposspronoun will use \pluralposspronoun, otherwise it will use \personposspronoun.

```
  \newcommand*{\peopleposspronoun}{%
    \ifnum\c@people>1\relax
       \pluralposspronoun
    \else
       \@get@firstperson{\@thisperson}%
       \personposspronoun[\@thisperson]%
    \fi
  }
```

opleposspronoun As above, but first letter in upper case

```
  \newcommand*{\Peopleposspronoun}{%
    \ifnum\c@people>1\relax
       \expandafter\MakeUppercase\pluralposspronoun
    \else
       \@get@firstperson{\@thisperson}%
       \Personposspronoun[\@thisperson]%
    \fi
  }
```

\personchild Display this person's relationship to their parent (i.e. son or daughter).

```
  \newcommand*{\personchild}[1][anon]{%
    \@ifundefined{person@#1@gender}%
    {%
      \PackageError{person}{Person ‘#1’ has not been defined}{}%
    }%
    {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \csname\@gender child\endcsname
    }%
  }
```

\Personchild As above, but make first letter uppercase.

```
  \newcommand*{\Personchild}[1][anon]{%
    \@ifundefined{person@#1@gender}%
    {%
      \PackageError{person}{Person ‘#1’ has not been defined}{}%
    }%
    {%
      \edef\@gender{\csname person@#1@gender\endcsname}%
      \expandafter\expandafter\expandafter\MakeUppercase
        \csname\@gender child\endcsname
    }%
```

```
}
```

\peoplechild If there is more than one person, \peoplechild will use \malechildren (if all male), \femalechildren (if all female) or \pluralchild (if mixed), otherwise it will use \personchild.

```
\newcommand*{\peoplechild}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\malechildren}%
      {\ifallfemale{\femalechildren}{\pluralchild}}%
  \else
    \@get@firstperson{\@thisperson}%
    \personchild[\@thisperson]%
 \fi
}
```

\Peoplechild As above but first letter is made uppercase.

```
\newcommand*{\Peoplechild}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\expandafter\MakeUppercase\malechildren}%
      {\ifallfemale
        {\expandafter\MakeUppercase\femalechildren}
        {\expandafter\MakeUppercase\pluralchild}}%
  \else
    \@get@firstperson{\@thisperson}%
    \Personchild[\@thisperson]%
 \fi
}
```

\personparent Display this person's relationship to their child (i.e. father or mother).

```
\newcommand*{\personparent}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender parent\endcsname
  }%
}
```

\Personparent As above, but make the first letter uppercase.

```
\newcommand*{\Personparent}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
```

```
      \expandafter\expandafter\expandafter\MakeUppercase
        \csname\@gender parent\endcsname
    }%
  }
```

\peopleparent If there is more than one person, `\peopleparent` will use `\pluralparent`, otherwise it will use `\personparent`.

```
\newcommand*{\peopleparent}{%
  \ifnum\c@people>1\relax
     \pluralparent
  \else
     \@get@firstperson{\@thisperson}%
     \personparent[\@thisperson]%
  \fi
}
```

\Peopleparent As above, but make first letter uppercase.

```
\newcommand*{\Peopleparent}{%
  \ifnum\c@people>1\relax
     \expandafter\MakeUppercase\pluralparent
  \else
     \@get@firstperson{\@thisperson}%
     \Personparent[\@thisperson]%
  \fi
}
```

\personsibling Display this person's relationship to their siblings (i.e. brother or sister).

```
\newcommand*{\personsibling}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \csname\@gender sibling\endcsname
  }%
}
```

\Personsibling Display this person's relationship to their siblings (i.e. brother or sister).

```
\newcommand*{\Personsibling}[1][anon]{%
  \@ifundefined{person@#1@gender}%
  {%
    \PackageError{person}{Person ‘#1’ has not been defined}{}%
  }%
  {%
    \edef\@gender{\csname person@#1@gender\endcsname}%
    \expandafter\expandafter\expandafter\MakeUppercase
      \csname\@gender sibling\endcsname
  }%
}
```

\peoplesibling If there is more than one person, \peoplesibling will use \malesiblings (if all male), \femalesiblings (if all female) or \pluralsibling (if mixed), otherwise it will use \personsibling.

```
\newcommand*{\peoplesibling}{%
  \ifnum\c@people>1\relax
    \ifallmale
      {\malesiblings}%
      {\ifallfemale{\femalesiblings}{\pluralsibling}}%
  \else
    \@get@firstperson{\@thisperson}%
    \personsibling[\@thisperson]%
 \fi
}
```

\persongender Displays the given person's gender (\malename or \femalename).

```
\newcommand*{\persongender}[1]{%
   \ifmale{#1}{\malename}{\femalename}%
}
```

10.7 Extracting Information

getpersongender Gets person's gender and stores in first argument which must be a control sequence.

```
\newcommand*{\getpersongender}[2]{%
   \ifmale{#2}{\let#1\malename}{\let#1\femalename}%
}
```

\getpersonname Gets person's name and stores in first argument which must be a control sequence.

```
\newcommand*{\getpersonname}[2]{%
   \ifpersonexists{#2}%
   {%
     \expandafter\let\expandafter#1\csname person@#2@name\endcsname
   }%
   {%
      \PackageError{person}{Person ‘#2’ doesn’t exist}{}%
   }%
}
```

tpersonfullname Gets person's full name and stores in first argument which must be a control sequence.

```
\newcommand*{\getpersonfullname}[2]{%
   \ifpersonexists{#2}%
   {%
     \expandafter
       \let\expandafter#1\csname person@#2@fullname\endcsname
   }%
   {%
      \PackageError{person}{Person ‘#2’ doesn’t exist}{}%
```

```
    }%
}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.

Change History

- \dtlswapentriesincurrentrow: new 146
- \long@addto@envbody: new 6
- \long@collect@@body: new 7
- \long@push@begins: new 7

2.11 (2012-09-25)

- \@dtl@updatekeys: remove unwanted space 129
- \@dtlgetrowindex: new 151
- \DTLaddcolumn: new 128
- \dtldisplayvalign: new 177
- \dtlgetrowforvalue: new 142
- \DTLgetrowindex: new 150
- \dtlgetrowindex: new 151
- \dtlupdateentryincurrentrow: new 148

2.12 (2012-11-30)

- \@DTLforeach: fixed bug in \DTLiflastrow 160
- \@dtl@construct@getintfrac: switched to \ifdefempty 17
- \@sDTLforeach: fixed bug in \DTLiflastrow 164
- \dtlifnumclosedbetween: fixed bug causing premature expansion 109
- \dtlifnumeq: fixed bug causing premature expansion 108
- \dtlifnumgt: fixed bug causing premature expansion 109
- \dtlifnumlt: fixed bug causing premature expansion 109
- \dtlifnumopenbetween: fixed bug causing premature expansion 109

2.13 (2013-01-15)

- \@DTLnewrow: fixed typo in \PackageError 121
- \@dtl@getsortdirection: removed spurious space 207
- \@dtl@setheaderforindex: removed spurious space 132
- \@dtlnumbernull: new 138
- \@dtlstringnull: new 138
- General: added datagidx package 229
- math: changed \newcommand to \providecommand 116
- \dtl@setcharcode: change from check for space and tilde to check for \@dtl@wordbreak ¯¯59
- \dtl@setlccharcode: changed to test for \@dtl@wordbreak instead of \space and tilde ¯¯66
- fixed bug where characters without a lower case equivalent were all considered equal ¯¯66
- \dtl@testifsubstring: now using \dtl@setwordbreaksnohyphens to deal with spaces ¯¯84
- \dtl@teststartswith: now using \dtl@setwordbreaksnohyphens to deal with spaces ¯¯86
- \DTLaddentryforrow: removed spurious space 172
- \DTLdisplaydb: removed spurious space 178
- \DTLdisplaylongdb: removed spurious space 180
- \DTLforeachkeyinrow: changed to use \@dtlstringnull and \@dtlnumbernull ¯¯174
- \DTLgcleardb: new 120
- \DTLgdeletedb: new 119
- \DTLgnewdb: new 119
- \dtlicompare: now using \dtl@setwordbreaksnohyphens to deal with spaces ¯¯67
- \dtlparsewords: new 79
- \DTLrawmap: removed spurious space 227
- \DTLsaverawdb: new 212
- \dtlsort: new 200
- \ifDTLnewdbonload: new 219

2.14 (2013-06-28)

- \@dtl@updatekeys: expand value before testing if it's empty 129
- General: added calc library requirement 409
- \datagidx@add@term: added \postnewtermhook 295
- \datagidx@style@gloss: removed spurious see also line 269
- \datagidxdb: new 299
- \dtl@constructminorticklist: replaced \FPsub with \dtlsub etc 433
- \dtl@constructticklist: replaced \FPsub with \dtlsub etc 431
- \dtl@constructticklistwithgap: replaced \FPadd with \dtladd 432
- \dtl@constructticklistwithgapex: replaced \FPadd with \dtladd and changed third argument to minimum gap width (in data co-ordinates) ¯¯434

- \dtl@constructticklistwithgapz: replaced \FPadd with \dtladd etc 432
- \dtl@getbounds: changed \FPifgt to \dtlifnumgt 430
- \DTLgidxForeachEntry: Added optional argument when using \DTLforeach 322
- \DTLpar: changed to \let 116
- \DTLplot: replaced \FPround with \dtlround 420
  - replaced \FPsub etc with \dtlsub etc ¯¯418
- \ifnewtermfield: new 296
- \newtermfield: new 296
- \printterms@condition: new 318

2.15 (2013-07-10)

- \@dtlminforkeys: Replaced \ifstrempty with \ifdefempty 195
- General: added afterpage as a required package 229
- \datagidx@target: fixed page breaking bug 303
- \datagidx@write@usedentry: Added check for page counter 308
- \do@locrange: removed spurious space 279
- \DTLaddtoplotlegend: Used \xdef instead of \edef as may be scoped. 434
- \DTLgidxForeachEntry: expanded and sanitize locations before comparing them 322
- \dtlplothandlermark: new 413
- \DTLprotectedsaverawdb: new 215
- \DTLsaverawdb: added \dtllastloadeddb 214
- \DTLsavetexdb: added \dtllastloadeddb 212
- \loadgidx: new 281
- nowarn: new 240

2.16 (2013-08-16)

- \@dtl@checknumerical: Moved empty check 46
- \@dtl@updatekeys: reverted to not expanding value (2.14 change causes an error with fragile commands). Fix now in \@dtl@checknumerical ¯¯129

2.17 (2013-08-29)

- \DTLfetch: new 142
- \edtlgetrowforvalue: new 142

2.18 (2013-09-06)

- \DTLpar: changed back to a robust command 116
- \dtlsort: added check for existence of keys listed in sort criteria 200

2.19 (2014-01-17)

- \@datagidx@use@entry: renamed \datagidx@use@entry to \@datagidx@use@entry and removed redundant field argument ¯¯306
- \@dtl@resetdostartrow: switched to \dtldisplaycr 180
- \dtldisplaycr: new 177
- \DTLdisplaydb: switched to \dtldisplaycr 179
- \glsaddall: Fixed bug in database reference 311, 312

2.20 (2014-02-03)

- \@DTLforeach: change \gdef to \xdef 159
- \@dtl@assign: Added check for empty argument 136
- \@dtl@assigncmd: Stored db label in \@dtl@dbname 137
- \@sDTLforeach: Added missing \@dtl@dbname assignment 164
- \DTLassignfirstmatch: new 135
- \DTLifnullorempty: new 139
- \DTLloaddbtex: new 217
- \xDTLassignfirstmatch: new 136

2.21 (2014-03-08)

- \@gDTLforeachbibentry: new 330
- \@sgDTLforeachbibentry: new 331
- \datagidx@parse@location: replaced \ifstrequal with \ifdefequal 275
- \dtl@g@gathervalues: new 141
- \dtl@gathervalues: fixed bug that ignore row tok argument 141
- \DTLdisplaylongdb: added missing \dtldisplayafterhead 182
  - fixed location of \dtldisplaystarttab 182
  - moved misplaced \dtldisplayafterhead 183
- \DTLforeachbibentry: fixed bug in starred version 329
- \gDTLforeachbibentry: new 330
- \gDTLformatbibentry: new 328

2.22 (2014-06-10)

- \@dtl@chopexcessfrac: new 23